

---

**Pacific Northwest  
National Laboratory**

Operated by Battelle for the  
U.S. Department of Energy

## **Updated User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes**

### **Volume 3: Utility Codes**

P. W. Eslinger  
R. L. Aaberg  
C. Arimescu  
C. A. Lopresti

T. B. Miley  
W. E. Nichols  
D. L. Streng

September 2006

Prepared for the U.S. Department of Energy  
under Contract DE-AC05-76RL01830



## DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.** Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY

*operated by*

BATTELLE

*for the*

UNITED STATES DEPARTMENT OF ENERGY

*under Contract DE-AC05-76RL01830*

Printed in the United States of America

Available to DOE and DOE contractors from the  
Office of Scientific and Technical Information,

P.O. Box 62, Oak Ridge, TN 37831-0062;

ph: (865) 576-8401

fax: (865) 576-5728

email: [reports@adonis.osti.gov](mailto:reports@adonis.osti.gov)

Available to the public from the National Technical Information Service,  
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161

ph: (800) 553-6847

fax: (703) 605-6900

email: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov)

online ordering: <http://www.ntis.gov/ordering.htm>



This document was printed on recycled

**Updated User Instructions for the Systems  
Assessment Capability, Rev. 1, Computer Codes**

**Volume 3: Utility Codes**

P. W. Eslinger  
R. L. Aaberg  
C. Arimescu  
C. A. Lopresti  
T. B. Miley  
W. E. Nichols  
D. L. Streng

September 29, 2006

Prepared for  
the U.S. Department of Energy  
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory  
Richland, Washington 99352



## Contents

1.0	Introduction and Background.....	1
1.1	Overview of the SAC Systems Code.....	1
1.2	Purpose of This Document .....	2
1.3	Overview of the Utility Codes.....	3
1.4	General Rules for Reading Keyword Descriptions .....	6
2.0	ANIMATE – Animation of Impact Code Results.....	7
2.1	Overview .....	7
2.1.1	Location in the Processing Sequence .....	7
2.1.2	How the Code Is Invoked.....	7
2.1.3	Memory Requirements .....	8
2.2	File Definitions.....	8
2.2.1	Input Files.....	8
2.2.2	Output Files .....	9
2.3	Top-Level Keywords for Every Run of ANIMATE .....	10
2.3.1	CATEGORY Keyword for ANIMATE .....	10
2.3.2	DEBUG Keyword for ANIMATE .....	10
2.3.3	END Keyword for ANIMATE.....	10
2.3.4	LEGEND Keyword for ANIMATE .....	11
2.3.5	LOCATION Keyword for ANIMATE.....	11
2.3.6	PICK_REL Keyword for ANIMATE .....	11
2.3.7	REPORT Keyword for ANIMATE.....	11
2.3.8	TIMES Keyword for ANIMATE .....	12
2.3.9	TITLE Keyword for ANIMATE.....	12
2.3.10	USER Keyword for ANIMATE.....	12
2.3.11	VERBOSE Keyword for ANIMATE.....	12
2.4	Keywords and Keyword File Specific to CULTURE Data.....	13
2.4.1	FILE Keyword.....	13
2.4.2	MAP Keyword .....	13
2.4.3	Example Keyword File for CULTURE Map Data.....	13
2.5	Keywords and Keyword File Specific to ECDA Data .....	14
2.5.1	ANALYTE Keyword .....	14
2.5.2	FILE Keyword.....	14
2.5.3	MEDIA Keyword.....	15
2.5.4	UNITS Keyword .....	16
2.5.5	Example Keyword File for ECDA Data.....	16
2.6	Keywords and Keyword File Specific to ECEM Data .....	17
2.6.1	ANALYTE Keyword .....	17
2.6.2	FILE Keyword.....	17
2.6.3	SOILTYPE Keyword .....	17
2.6.4	SOLUTION Keyword.....	18
2.6.5	SPECIES Keyword .....	18
2.6.6	UNITS Keyword .....	19
2.6.7	Example Keyword File for ECEM Data .....	19

2.7	Keywords and Keyword File Specific to FCDA Data .....	20
2.7.1	ANALYTE Keyword .....	20
2.7.2	FILE Keyword.....	20
2.7.3	SOIL Keyword .....	20
2.7.4	SPECIES Keyword .....	21
2.7.5	UNITS Keyword .....	21
2.7.6	Example Keyword File for FCDA Data.....	21
2.8	Keywords and Keyword File Specific to HUMAN Data.....	22
2.8.1	ANALYTE Keyword.....	22
2.8.2	ANALYTE Keyword .....	23
2.8.3	FILE Keyword.....	23
2.8.4	SOLUTION Keyword .....	24
2.8.5	UNITS Keyword .....	25
2.8.6	Example Keyword File for HUMAN Data .....	25
3.0	Back_MASS2 – Columbia River Background Data Generation .....	27
3.1	Overview .....	27
3.1.1	Location in the Processing Sequence .....	28
3.1.2	How the Code Is Invoked.....	29
3.1.3	Memory Requirements .....	29
3.2	File Definitions.....	29
3.2.1	Input Files.....	29
3.2.2	Output Files .....	30
3.3	Keyword Definitions for Back_MASS2 .....	31
3.3.1	ANALYTE Keyword for Back_MASS2 .....	31
3.3.2	DISSOLVED Keyword for Back_MASS2 .....	31
3.3.3	END Keyword for Back_MASS2 .....	32
3.3.4	PERIOD Keyword for Back_MASS2.....	32
3.3.5	REALIZAT Keyword for Back_MASS2.....	32
3.3.6	RIVER Keyword for Back_MASS2 .....	33
3.3.7	SEDIMENT Keyword for Back_MASS2 .....	33
3.3.8	TITLE Keyword for Back_MASS2 .....	33
3.3.9	USER Keyword for Back_MASS2 .....	33
4.0	CONSUME – Ecological Species Consumption Data.....	35
4.1	Overview .....	35
4.1.1	Location in the Processing Sequence .....	35
4.1.2	Memory Requirements .....	35
4.2	File Definitions.....	35
4.2.1	Input Files.....	35
4.2.2	Output Files .....	36
4.3	Code Execution .....	37
5.0	CRGRAB – Columbia River Data Grabber .....	39
5.1	Overview .....	39
5.1.1	Location in the Processing Sequence .....	39
5.1.2	How the Code Is Invoked.....	39
5.1.3	Memory Requirements .....	39
5.2	File Definitions.....	39

5.2.1	Input Files.....	39
5.2.2	Output Files.....	41
5.3	Keyword Definitions for CRGRAB.....	42
5.3.1	ANALYTE Keyword for CRGRAB.....	42
5.3.2	BASEPATH Keyword for CRGRAB.....	43
5.3.3	DEBUG Keyword for CRGRAB.....	43
5.3.4	END Keyword for CRGRAB.....	44
5.3.5	EXTRACT Keyword for CRGRAB.....	44
5.3.6	GRABPATH Keyword for CRGRAB.....	44
5.3.7	IGNORE Keyword for CRGRAB.....	44
5.3.8	LOCATION Keyword for CRGRAB.....	45
5.3.9	MASS2GRID Keyword for CRGRAB.....	45
5.3.10	ONLY Keyword for CRGRAB.....	45
5.3.11	REALIZAT Keyword for CRGRAB.....	46
5.3.12	TIME Keyword for CRGRAB.....	46
5.3.13	TITLE Keyword for CRGRAB.....	46
5.3.14	USER Keyword for CRGRAB.....	47
5.3.15	VERBOSE Keyword for CRGRAB.....	47
6.0	CSCALE – Concentration Data Scaling.....	49
6.1	Overview.....	49
6.1.1	Location in the Processing Sequence.....	49
6.1.2	How the Code Is Invoked.....	49
6.1.3	Memory Requirements.....	50
6.2	File Definitions.....	50
6.2.1	Input Files.....	50
6.2.2	Output Files.....	51
6.3	Keywords Definitions for CSCALE.....	52
6.3.1	ANALYTE Keyword for CSCALE.....	52
6.3.2	DEBUG Keyword for CSCALE.....	52
6.3.3	END Keyword for CSCALE.....	53
6.3.4	EXECUTE Keyword for CSCALE.....	53
6.3.5	FILE Keyword for CSCALE.....	53
6.3.6	REPORT Keyword for CSCALE.....	54
6.3.7	SCALE Keyword for CSCALE.....	54
6.3.8	TITLE Keyword for CSCALE.....	55
6.3.9	USER Keyword for CSCALE.....	55
7.0	ECDA – Environmental Concentration Data Accumulator Setup.....	57
7.1	Overview.....	57
7.1.1	Location in the Processing Sequence.....	57
7.1.2	How the Code Is Invoked.....	57
7.1.3	Memory Requirements.....	57
7.2	File Definitions.....	57
7.2.1	Input Files for ECDA.....	58
7.2.2	Output Files for ECDA.....	59
7.3	Keyword Definitions for ECDA.....	60
7.3.1	ANALYTE Keyword for ECDA.....	60

7.3.2	DEBUG Keyword for ECDA.....	61
7.3.3	DILUTE Keyword for ECDA .....	61
7.3.4	ECHO Keyword for ECDA.....	61
7.3.5	END Keyword for ECDA .....	62
7.3.6	FILE Keyword for ECDA .....	62
7.3.7	FILLECDA Keyword for ECDA .....	63
7.3.8	KDSOIL Keyword in ECDA .....	64
7.3.9	LOCATION Keyword for ECDA .....	65
7.3.10	PERIOD Keyword for ECDA .....	66
7.3.11	REALIZAT Keyword for ECDA .....	67
7.3.12	TITLE Keyword for ECDA .....	67
7.3.13	USER Keyword for ECDA .....	67
8.0	ECDA_ASCII – Environmental Concentration Data Accumulator File Conversions.....	69
8.1	Overview .....	69
8.1.1	Location in the Processing Sequence .....	69
8.1.2	How the Code Is Invoked.....	69
8.1.3	Memory Requirements .....	69
8.2	File Definitions.....	69
8.2.1	Input Files.....	69
8.2.2	Output Files .....	70
8.3	Code Execution .....	70
9.0	ECDA_BACK – Environmental Concentration Fill-In.....	73
9.1	Overview .....	73
9.1.1	Location in the Processing Sequence .....	73
9.1.2	How the Code Is Invoked.....	73
9.1.3	Memory Requirements .....	73
9.2	File Definitions.....	73
9.2.1	Input Files for ECDA_BACK .....	74
9.2.2	Output Files for ECDA_BACK .....	74
9.3	Keyword Definitions for ECDA_BACK.....	75
9.3.1	ANALYTE Keyword for ECDA_BACK.....	75
9.3.2	END Keyword for ECDA_BACK .....	75
9.3.3	FILE Keyword for ECDA_BACK .....	75
9.3.4	REPORT Keyword for ECDA_BACK .....	76
9.3.5	TITLE Keyword for ECDA_BACK .....	76
9.3.6	USER Keyword for ECDA_BACK .....	77
9.3.7	VERBOSE Keyword for ECDA_BACK .....	77
9.3.8	YEAR Keyword for ECDA_BACK.....	77
10.0	FCDA_ASCII – Food Concentration Data File Conversions .....	79
10.1	Overview .....	79
10.1.1	Location in the Processing Sequence .....	79
10.1.2	How the Code Is Invoked.....	79
10.1.3	Memory Requirements .....	79
10.2	File Definitions.....	79
10.2.1	Input Files.....	79
10.2.2	Output Files .....	79



10.3	Code Execution .....	80
11.0	FILLECDA – Environmental Concentration Data File Modifications .....	81
11.1	Overview .....	81
11.1.1	Location in the Processing Sequence .....	81
11.1.2	How the Code Is Invoked .....	81
11.1.3	Memory Requirements .....	81
11.2	File Definitions.....	81
11.2.1	Input Files.....	82
11.2.2	Output Files.....	82
11.3	Keyword Definitions for FILLECDA .....	83
11.3.1	DATA Keyword for FILLECDA .....	83
11.3.2	END Keyword for FILLECDA .....	84
11.3.3	FILE Keyword for FILLECDA.....	84
11.3.4	REALIZAT Keyword for FILLECDA.....	84
11.3.5	REPORT Keyword for FILLECDA .....	84
11.3.6	TITLE Keyword for FILLECDA .....	85
11.3.7	USER Keyword for FILLECDA .....	85
12.0	FILLFCDA – Food Concentration Data File Modifications.....	87
12.1	Overview .....	87
12.1.1	Location in the Processing Sequence .....	87
12.1.2	How the Code Is Invoked.....	87
12.1.3	Memory Requirements .....	87
12.2	File Definitions.....	87
12.2.1	Input Files.....	88
12.2.2	Output Files.....	88
12.3	Keyword Definitions for FILLFCDA .....	89
12.3.1	DATA Keyword for FILLFCDA .....	89
12.3.2	END Keyword for FILLFCDA .....	90
12.3.3	FILE Keyword for FILLFCDA.....	90
12.3.4	REALIZAT Keyword for FILLFCDA.....	91
12.3.5	REPORT Keyword for FILLFCDA .....	91
12.3.6	TITLE Keyword for FILLFCDA .....	91
12.3.7	USER Keyword for FILLFCDA .....	91
13.0	GETTRACE – Human Risk Extractor.....	93
13.1	Overview .....	93
13.1.1	Location in the Processing Sequence .....	93
13.1.2	How the Code Is Invoked.....	93
13.1.3	Memory Requirements .....	93
13.2	File Definitions.....	93
13.2.1	Input Files.....	93
13.2.2	Output Files.....	94
13.3	Keyword Definitions for the GETTRACE Code .....	95
13.3.1	ANALYTE Keyword for GETTRACE.....	95
13.3.2	END Keyword for GETTRACE .....	97
13.3.3	FACTOR Keyword for GETTRACE.....	97
13.3.4	FILE Keyword for GETTRACE .....	98

13.3.5	HEADER Keyword for GETTRACE .....	98
13.3.6	REPORT Keyword for GETTRACE .....	98
13.3.7	USER Keyword for GETTRACE .....	99
14.0	GWGRAB – Groundwater Transport Module Data Extractor.....	101
14.1	Overview .....	101
14.1.1	Location in the Processing Sequence .....	101
14.1.2	How the Code Is Invoked.....	101
14.1.3	Memory Requirements .....	101
14.2	File Definitions.....	101
14.2.1	Input Files.....	101
14.2.2	Output Files .....	102
14.3	Keyword Descriptions for the GWGRAB Control Keyword File .....	102
14.3.1	ANALYTE Keyword for GWGRAB.....	102
14.3.2	END Keyword for GWGRAB .....	102
14.3.3	EXEDIR Keyword for GWGRAB .....	102
14.3.4	EXTRACT Keyword for GWGRAB .....	103
14.3.5	PATH Keyword for GWGRAB .....	103
14.3.6	REALIZAT Keyword for GWGRAB .....	103
14.3.7	OUTPUT Keyword for GWGRAB.....	104
14.3.8	TIME Keyword for GWGRAB .....	104
14.4	Example Cases for GWGRAB.....	104
14.4.1	GWGRAB Example – Check Run Status for CFEST.....	104
14.4.2	GWGRAB Example – Extract CFEST Results.....	105
15.0	HIGHDOSE – Dose Extraction for ECEM.....	107
15.1	Overview .....	107
15.1.1	Location in the Processing Sequence .....	107
15.1.2	How the Code Is Invoked.....	107
15.1.3	Memory Requirements .....	107
15.2	File Definitions.....	107
15.2.1	Input Files.....	107
15.2.2	Output Files .....	109
15.3	Keyword Definitions for HIGHDOSE .....	110
15.3.1	ANALYTE Keyword for HIGHDOSE .....	110
15.3.2	END Keyword for HIGHDOSE.....	110
15.3.3	FILE Keyword for HIGHDOSE.....	110
15.3.4	LOCATION Keyword for HIGHDOSE.....	111
15.3.5	MEMORY Keyword for HIGHDOSE .....	111
15.3.6	REALIZAT Keyword for HIGHDOSE .....	111
15.3.7	REPORT Keyword for HIGHDOSE.....	112
15.3.8	SOIL Keyword for HIGHDOSE .....	112
15.3.9	SOLUTION Keyword for HIGHDOSE .....	112
15.3.10	TIME Keyword for HIGHDOSE .....	113
15.3.11	TITLE Keyword for HIGHDOSE.....	113
15.3.12	USER Keyword for HIGHDOSE.....	113
15.3.13	VERBOSE Keyword for HIGHDOSE.....	114
16.0	HIGHIMPACT – Maximum Impact Extraction for HUMAN.....	115

16.1	Overview .....	115
16.1.1	Location in the Processing Sequence .....	115
16.1.2	How the Code Is Invoked.....	115
16.1.3	Memory Requirements .....	115
16.2	File Definitions.....	115
16.2.1	Input Files.....	115
16.2.2	Output Files .....	116
16.3	Keyword Definitions for the HIGHIMPACT Code .....	117
16.3.1	ANA_ID Keyword for HIGHIMPACT.....	117
16.3.2	END Keyword for HIGHIMPACT .....	117
16.3.3	FILE Keyword for HIGHIMPACT .....	117
16.3.4	LOC_ID Keyword for HIGHIMPACT .....	118
16.3.5	REALIZAT Keyword for HIGHIMPACT .....	118
16.3.6	REPORT Keyword for HIGHIMPACT .....	119
16.3.7	R_TYPE Keyword for HIGHIMPACT.....	119
16.3.8	S_TYPE Keyword for HIGHIMPACT .....	119
16.3.9	TIME Keyword for HIGHIMPACT.....	121
16.3.10	TITLE Keyword for HIGHIMPACT .....	121
16.3.11	USER Keyword for HIGHIMPACT .....	121
16.3.12	VERBOSE Keyword for HIGHIMPACT .....	121
17.0	HIGHMEDIA – Extraction of Maximum Media Concentrations.....	123
17.1	Overview .....	123
17.1.1	Location in the Processing Sequence .....	123
17.1.2	How the Code Is Invoked.....	123
17.1.3	Memory Requirements .....	123
17.2	File Definitions.....	123
17.2.1	Input Files.....	124
17.2.2	Output Files .....	124
17.3	Keyword Definitions for HIGHMEDIA .....	125
17.3.1	Keywords in the Initial Section for HIGHMEDIA .....	125
17.3.2	Keywords for Specific Cases for HIGHMEDIA.....	126
17.3.3	Concluding (END) Keyword for HIGHMEDIA .....	130
18.0	HTWOS_TDP – Tank Inventory Data Processor .....	131
18.1	Overview .....	131
18.1.1	Location in the Processing Sequence .....	132
18.1.2	How the Code Is Invoked.....	132
18.2	File Definitions.....	132
18.2.1	Control Data File.....	132
18.2.2	HTWOS Annual Data Files.....	133
18.2.3	Radionuclide Decay Chain Library .....	134
18.3	Output Files .....	134
19.0	HTWOS_TLDP – Hanford Tank Leak Data Processing .....	137
19.1	Overview .....	137
19.1.1	Location in the Processing Sequence .....	137
19.1.2	How the Code Is Invoked.....	137
19.1.3	Processing Assumptions.....	137

19.2	File Definitions.....	137
19.2.1	Input Files.....	137
19.2.2	Output Files.....	139
20.0	Imp_Med – Median Values for Impact Codes.....	141
20.1	Overview .....	141
20.1.1	Location in the Processing Sequence .....	141
20.1.2	How the Code Is Invoked.....	141
20.1.3	Memory Requirements .....	141
20.2	Computational Sequence.....	141
20.2.1	IMP_STEP1 Execution .....	141
20.2.2	STOCHASTIC Execution .....	142
20.2.3	IMP_STEP2 Execution .....	142
21.0	INGRAB – Inventory Data Extraction.....	143
21.1	Overview .....	143
21.1.1	Location in the Processing Sequence .....	143
21.1.2	How the Code Is Invoked.....	143
21.1.3	Memory Requirements .....	143
21.2	File Definitions.....	143
21.2.1	Input Files.....	144
21.2.2	Output Files.....	144
21.3	Keyword Definitions INGRAB.....	145
21.3.1	ANALYTE Keyword for INGRAB .....	145
21.3.2	END Keyword for INGRAB.....	145
21.3.3	EXECUTE Keyword for INGRAB.....	145
21.3.4	FILE Keyword for INGRAB.....	145
21.3.5	HALFLIFE Keyword for INGRAB .....	146
21.3.6	REALIZAT Keyword for INGRAB.....	146
21.3.7	REPORT Keyword for INGRAB.....	146
21.3.8	SHOW Keyword for INGRAB .....	147
21.3.9	SITE Keyword for INGRAB.....	147
21.3.10	TITLE Keyword for INGRAB.....	147
21.3.11	TYPE Keyword for INGRAB .....	148
21.3.12	USER Keyword for INGRAB.....	148
21.3.13	YEARS Keyword for INGRAB.....	148
22.0	INGRES – Inventory Data Extraction.....	149
22.1	Overview .....	149
22.1.1	Location in the Processing Sequence .....	149
22.1.2	How the Code Is Invoked.....	149
22.1.3	Memory Requirements .....	149
22.2	File Definitions.....	149
22.2.1	Input Files.....	149
22.2.2	Output Files.....	150
22.3	Keyword Definitions INGRES.....	151
22.3.1	ANALYTE Keyword for INGRES .....	151
22.3.2	END Keyword for INGRES.....	151
22.3.3	FILE Keyword for INGRES.....	151

22.3.4	REPORT Keyword for INGRES.....	152
22.3.5	TITLE Keyword for INGRES.....	152
22.3.6	USER Keyword for INGRES.....	152
22.3.7	VERBOSE Keyword for INGRES.....	152
22.3.8	WASTE Keyword for INGRES.....	153
22.3.9	YEAR Keyword for INGRES.....	153
23.0	INPROC – Inventory Preprocessing.....	155
23.1	Overview.....	155
23.1.1	Location in the Processing Sequence.....	155
23.1.2	How the Code Is Invoked.....	155
23.1.3	Memory Requirements.....	156
23.1.4	Inventory Estimation Rules.....	156
23.1.5	Statistical Rules.....	156
23.2	File Definitions.....	156
23.2.1	Input Files.....	157
23.2.2	Output Files.....	160
23.3	Keyword Definitions for INPROC.....	162
23.3.1	ANALYTE Keyword for INPROC.....	162
23.3.2	ATTRIBUTE Keyword for INPROC.....	163
23.3.3	END Keyword for INPROC.....	163
23.3.4	ESTIMATE Keyword for INPROC.....	164
23.3.5	FILE Keyword for INPROC.....	165
23.3.6	PERIOD Keyword for INPROC.....	166
23.3.7	STATRULE Keyword for INPROC.....	166
23.3.8	SURROGATE Keyword for INPROC.....	168
23.3.9	SUM Keyword for INPROC.....	169
23.3.10	VERBOSE Keyword for INPROC.....	169
24.0	Inv_Med – Median Values for Inventory.....	171
24.1	Overview.....	171
24.1.1	Location in the Processing Sequence.....	171
24.1.2	How the Code Is Invoked.....	171
24.1.3	Memory Requirements.....	171
24.2	Computational Sequence.....	171
24.2.1	INV_STEP1 Execution.....	171
24.2.2	STOCHASTIC Execution.....	172
24.2.3	INV_STEP2 Execution.....	172
25.0	Inventory Database.....	175
25.1	Overview.....	175
25.1.1	Location in the Processing Sequence.....	175
25.1.2	Memory Requirements.....	175
25.2	Database Structure.....	175
25.3	Importing Data into the Inventory Database.....	178
25.3.1	Importing Record Data.....	178
25.3.2	Importing SIM Files.....	178
25.3.3	Importing HTWOS Files.....	178
25.4	Exporting Records from the Inventory Database.....	178

25.4.1	Query Create Inventory_Data for Selected Analytes .....	178
25.4.2	Query Create_INPROC_Onsite .....	179
25.4.3	Query Append HTWOS combined to INPROC_onsite .....	179
25.4.4	Query Append Stored Mixed Waste to INPROC_onsite in site 218-E-RCRA .....	179
25.4.5	Query Create INPROC_Input .....	179
26.0	INVEXT – Inventory Data Extractor .....	181
26.1	Overview .....	181
26.1.1	Location in the Processing Sequence .....	181
26.1.2	How the Code Is Invoked .....	181
26.1.3	Memory Requirements .....	181
26.2	Data Files .....	181
26.2.1	Input Files .....	182
26.2.2	Output Files .....	184
26.3	Keyword Definitions for INVEXT .....	186
26.3.1	ANALYTE Keyword for INVEXT .....	186
26.3.2	END Keyword for INVEXT .....	186
26.3.3	FILE Keyword for INVEXT .....	187
26.3.4	LEAK Keyword for INVEXT .....	187
26.3.5	REPORT Keyword for INVEXT .....	187
26.3.6	TANK Keyword for INVEXT .....	188
26.3.7	TIME Keyword for INVEXT .....	188
26.3.8	WASTES Keyword for INVEXT .....	188
26.3.9	SITES Keyword for INVEXT .....	189
26.3.10	TITLE Keyword for INVEXT .....	189
26.3.11	USER Keyword for INVEXT .....	189
26.3.12	VERBOSE Keyword for INVEXT .....	189
26.3.13	VOLUME Keyword for INVEXT .....	190
27.0	INVLOT – Inventory Plotting .....	191
27.1	Overview .....	191
27.1.1	Location in the Processing Sequence .....	191
27.1.2	How the Code Is Invoked .....	191
27.1.3	Memory Requirements .....	191
27.2	File Definitions .....	191
27.2.1	Input Files .....	192
27.2.2	Output Files .....	193
27.3	Keyword Definitions for the INVLOT Code .....	193
27.3.1	Control Section Keywords for the INVLOT Code .....	194
27.3.2	CASE Section Keywords for the INVLOT Code .....	196
27.3.3	END Keyword for INVLOT .....	198
28.0	INVREPAR – Inventory Data Repartitioning .....	199
28.1	Overview .....	199
28.1.1	Location in the Processing Sequence .....	199
28.1.2	How the Code Is Invoked .....	199
28.1.3	Memory Requirements .....	199
28.2	Data Files .....	199
28.2.1	Input Files .....	200

28.2.2	Output Files .....	200
28.3	Keyword Definitions for INVREPAR.....	201
28.3.1	DESTINAT Keyword for INVREPAR.....	201
28.3.2	END Keyword for INVREPAR .....	201
28.3.3	FILE Keyword for INVREPAR .....	201
28.3.4	OS Keyword for INVREPAR .....	203
28.3.5	SOURCE Keyword for INVREPAR.....	203
28.3.6	USER Keyword for INVREPAR .....	203
28.3.7	VERBOSE Keyword for INVREPAR .....	203
29.0	INVSUM – Inventory Data Summations.....	205
29.1	Overview .....	205
29.1.1	Location in the Processing Sequence .....	205
29.1.2	How the Code Is Invoked.....	205
29.1.3	Memory Requirements .....	205
29.2	File Definitions.....	205
29.2.1	Input Files.....	206
29.2.2	Output Files .....	207
29.3	Keyword Definitions for INVSUM.....	208
29.3.1	ANALYTE Keyword for INVSUM.....	208
29.3.2	END Keyword for INVSUM .....	209
29.3.3	FILE Keyword for INVSUM .....	209
29.3.4	LIMITOUT Keyword for INVSUM .....	210
29.3.5	LIMITSIT Keyword for INVSUM.....	210
29.3.6	REPORT Keyword for INVSUM .....	210
29.3.7	STATISTI Keyword for INVSUM .....	211
29.3.8	TITLE Keyword for INVSUM.....	211
29.3.9	USER Keyword for INVSUM .....	211
29.3.10	VERBOSE Keyword for INVSUM .....	211
29.3.11	WASTE Keyword for INVSUM.....	211
29.3.12	YEAR Keyword for INVSUM.....	212
30.0	L3IDIFF – Differencing CFEST Nodal Fluid Sources .....	213
30.1	Overview .....	213
30.1.1	Location in the Processing Sequence .....	213
30.1.2	How the Code Is Invoked.....	213
30.1.3	Memory Requirements .....	213
30.2	File Definitions.....	213
30.2.1	Input Files.....	213
30.2.2	Output Files .....	214
31.0	MAKEU – Inventory Data Uranium Conversion .....	215
31.1	Overview .....	215
31.1.1	Location in the Processing Sequence .....	215
31.1.2	How the Code Is Invoked.....	215
31.1.3	Memory Requirements .....	215
31.2	File Definitions.....	215
31.2.1	Input Files.....	216
31.2.2	Output Files .....	216

31.3	Keyword Definitions for MAKEU .....	217
31.3.1	END Keyword for MAKEU .....	217
31.3.2	FILE Keyword for MAKEU .....	217
31.3.3	OS Keyword for MAKEU .....	218
31.3.4	OUTPUT Keyword for MAKEU .....	218
31.3.5	SUM Keyword for MAKEU .....	218
31.3.6	TITLE Keyword for MAKEU .....	219
31.3.7	USER Keyword for MAKEU .....	219
31.3.8	VERBOSE Keyword for MAKEU .....	219
32.0	PICK_HYDRO – Constrained Vadose Zone Parameters .....	221
32.1	Overview .....	221
32.1.1	Algorithms .....	221
32.1.2	Location in the Processing Sequence .....	222
32.1.3	How the Code Is Invoked .....	222
32.1.4	Memory Requirements .....	222
32.2	File Definitions .....	222
32.2.1	Input Files .....	222
32.2.2	Output Files .....	223
32.3	Keyword Definitions for PICK_HYDRO .....	224
32.3.1	BASIS Keyword for PICK_HYDRO .....	225
32.3.2	CASE Keyword for PICK_HYDRO .....	225
32.3.3	END Keyword for PICK_HYDRO .....	225
32.3.4	ENDCASE Keyword for PICK_HYDRO .....	226
32.3.5	ENDINIT Keyword for PICK_HYDRO .....	226
32.3.6	FILE Keyword for PICK_HYDRO .....	226
32.3.7	GOOD Keyword for PICK_HYDRO .....	227
32.3.8	ITERATE Keyword for PICK_HYDRO .....	227
32.3.9	METRIC_M Keyword for PICK_HYDRO .....	228
32.3.10	METRIC_V Keyword for PICK_HYDRO .....	228
32.3.11	RELAX Keyword for PICK_HYDRO .....	229
32.3.12	REPORT Keyword for PICK_HYDRO .....	230
32.3.13	SAMPLES Keyword for PICK_HYDRO .....	230
32.3.14	SEED Keyword for PICK_HYDRO .....	230
32.3.15	STOCHASTIC Keyword for PICK_HYDRO .....	231
32.3.16	TITLE Keyword for PICK_HYDRO .....	232
32.3.17	USER Keyword for PICK_HYDRO .....	232
32.3.18	VERBOSE Keyword for PICK_HYDRO .....	232
33.0	PRESTOMP – STOMP Input File Modification (timing info) .....	233
33.1	Overview .....	233
33.1.1	Time Stepping Refinement .....	233
33.1.2	Vadose Zone Wetted Area Scaling .....	233
33.1.3	Location in the Processing Sequence .....	234
33.1.4	How the Code Is Invoked .....	234
33.2	File Definitions .....	236
33.2.1	Input Files .....	236
33.2.2	Output Files .....	240



34.0	RELUP – STOMP Release Time Step Threshold Reprocessor .....	241
34.1	Overview .....	241
34.1.1	Location in the Processing Sequence .....	241
34.1.2	How the Code Is Invoked .....	241
34.2	File Definitions.....	241
34.2.1	Input Files.....	242
34.2.2	Output Files .....	242
35.0	RLGRAB – Extractions from the Release Module .....	243
35.1	Overview .....	243
35.1.1	Location in the Processing Sequence .....	243
35.1.2	How the Code Is Invoked.....	243
35.1.3	Memory Requirements .....	243
35.2	File Definitions.....	243
35.2.1	Input Files.....	243
35.2.2	Output Files .....	244
35.3	Keyword Definitions for RLGRAB .....	244
35.3.1	ANALYTE Keyword for RLGRAB .....	244
35.3.2	END Keyword for RLGRAB.....	244
35.3.3	OUTFILE Keyword for RLGRAB.....	245
35.3.4	PATHNAME Keyword for RLGRAB .....	245
35.3.5	PERIOD Keyword for RLGRAB.....	245
35.3.6	REALIZAT Keyword for RLGRAB.....	246
35.3.7	SITE Keyword for RLGRAB.....	246
35.3.8	SITEFILE Keyword for RLGRAB .....	246
35.3.9	TITLE Keyword for RLGRAB .....	246
35.3.10	VARIABLE Keyword for RLGRAB .....	247
35.3.11	WASTETYP Keyword for RLGRAB .....	247
35.3.12	YEARS Keyword for RLGRAB .....	248
36.0	SETRES – STOMP Restart File Handling.....	249
36.1	Overview .....	249
36.1.1	Location in the Processing Sequence .....	249
36.1.2	Memory Requirements .....	249
36.1.3	How the Code Is Invoked.....	249
36.2	File Definitions.....	249
37.0	SIMS – SIM Inventory Data Processing.....	251
37.1	Overview .....	251
37.1.1	Location in the Processing Sequence .....	251
37.1.2	Memory Requirements .....	251
37.1.3	How the Code Is Invoked.....	251
37.2	File Definitions.....	251
37.2.1	Input Files.....	251
37.2.2	Output Files .....	252
37.3	Procedure for Processing SIM files for Input to Inventory Database.....	254
38.0	STOCHASTIC – Stochastic Values Generation.....	257
38.1	Overview .....	257
38.1.1	Location in the Processing Sequence .....	257

38.1.2	How the Code Is Invoked .....	257
38.1.3	Memory Requirements .....	257
38.2	File Definitions.....	257
38.2.1	Input Files.....	258
38.2.2	Output Files .....	258
38.3	Keyword Definitions for the STOCHASTIC Code.....	259
38.3.1	DEBUG Keyword for STOCHASTIC .....	259
38.3.2	END Keyword for STOCHASTIC.....	260
38.3.3	REALIZAT Keyword for STOCHASTIC .....	260
38.3.4	REPORT Keyword for STOCHASTIC .....	260
38.3.5	SEED Keyword for STOCHASTIC.....	260
38.3.6	SINGLEKEY Keyword for STOCHASTIC .....	261
38.3.7	STOCHASTIC Keyword for STOCHASTIC .....	261
38.3.8	TITLE Keyword for STOCHASTIC.....	263
38.3.9	USER Keyword for STOCHASTIC.....	263
39.0	Sto_Med – Median Values for Stomp (Vadose Zone).....	265
39.1	Overview .....	265
39.1.1	Location in the Processing Sequence .....	265
39.1.2	How the Code Is Invoked.....	265
39.1.3	Memory Requirements .....	265
39.2	Computational Sequence.....	265
39.2.1	STO_STEP1 Execution.....	265
39.2.2	STOCHASTIC Execution .....	266
39.2.3	STO_STEP2 Execution.....	266
40.0	VOLEXT – Waste Stream Volume Summaries.....	267
40.1	Overview .....	267
40.1.1	Location in the Processing Sequence .....	267
40.1.2	How the Code Is Invoked.....	267
40.1.3	Memory Requirements .....	267
40.2	File Definitions.....	267
40.2.1	Input Files.....	268
40.2.2	Output Files .....	268
40.3	Keyword Definitions for the VOLEXT Code .....	269
40.3.1	DEBUG Keyword for VOLEXT.....	269
40.3.2	END Keyword for VOLEXT .....	269
40.3.3	FILE Keyword for VOLEXT .....	270
40.3.4	SITES Keyword for VOLEXT.....	270
40.3.5	TITLE Keyword for VOLEXT .....	270
40.3.6	USER Keyword for VOLEXT .....	271
40.3.7	VERBOSE Keyword for VOLEXT .....	271
40.3.8	YEARS Keyword for VOLEXT .....	271
41.0	VZGRAB – Vadose Zone Data Grabber .....	273
41.1	Overview .....	273
41.1.1	How the Code Is Invoked.....	273
41.1.2	Memory Requirements .....	273
41.2	File Definitions.....	273

41.2.1	Input Files.....	273
41.2.2	Output Files.....	274
41.3	Keyword Descriptions for VZGRAB.....	275
41.3.1	ANALYTE Keyword for VZGRAB.....	275
41.3.2	CUMULATIVE Keyword for VZGRAB.....	275
41.3.3	END Keyword for VZGRAB.....	275
41.3.4	EXTRACT Keyword for VZGRAB.....	275
41.3.5	LIST keyword for VZGRAB.....	276
41.3.6	NOHEADER Keyword for VZGRAB.....	276
41.3.7	PATH keyword for VZGRAB.....	277
41.3.8	OUTPUT keyword for VZGRAB.....	277
41.3.9	REALIZAT keyword for VZGRAB.....	277
41.3.10	SITE Keyword for VZGRAB.....	277
41.3.11	TIME keyword for VZGRAB.....	278
42.0	VZSWAP – Vadose Zone Data Replacement.....	279
42.1	Overview.....	279
42.1.1	Location in the Processing Sequence.....	279
42.1.2	How the Code Is Invoked.....	279
42.1.3	Memory Requirements.....	279
42.2	File Definitions.....	279
43.0	WTRTBL and RDWTRTBL – Hydraulic Head Extractor for CFEST Results.....	281
43.1	Overview.....	281
43.1.1	Location in the Processing Sequence.....	281
43.1.2	How the Code Is Invoked.....	281
43.2	File Structure.....	281
43.2.1	Input Files.....	281
43.2.2	Output Files.....	282
44.0	Keyword Language Syntax.....	285
44.1	Keyword Records.....	285
44.2	Continuation Records.....	285
44.3	Comment Records.....	286
44.4	Input Data Handling.....	286
44.4.1	Quote Strings.....	286
44.4.2	Data Separators.....	287
45.0	Stochastic Variable Generation.....	289
45.1	Keywords Defining Stochastic Variables.....	289
45.2	Probability Concepts.....	293
45.3	Probability Integral Transform Method.....	293
45.4	Stratified Sampling.....	294
45.5	Generation Algorithms.....	295
45.5.1	Algorithm for the Uniform Distribution.....	295
45.5.2	Algorithm for the Discrete Uniform Distribution.....	296
45.5.3	Algorithm for the Loguniform Distribution.....	296
45.5.4	Algorithms for the Triangular Distribution.....	297
45.5.5	Algorithms for the Normal Distribution.....	298
45.5.6	Algorithms for the Lognormal Distribution.....	299

45.5.7 Algorithms for the User-Defined Distribution .....	299
45.5.8 Algorithms for the Beta Distribution.....	300
45.5.9 Algorithms for the Log Ratio Distribution.....	301
46.0 References.....	303

## Figures

Figure 1.1 Module Information Flow for the SAC Rev. 1 Systems Code .....	2
Figure 8.1 Screen Image of an ECDA_ASCII Run .....	71
Figure 10.1 Screen Image of an FCDA_ASCII Run.....	80
Figure 25.1 Relationships between Inventory Database Tables .....	175

## Tables

Table 1.1 Overview of Utility Codes.....	3
Table 2.1 Data Types Used in the ANIMATE Utility Code.....	7
Table 2.2 Modifiers Associated with the CATEGORY Keyword in ANIMATE .....	10
Table 2.3 Modifiers Associated with the FILE Keyword in ANIMATE for CULTURE Data .....	13
Table 2.4 Example Keyword File for ANIMATE for CULTURE Upland Data .....	14
Table 2.5 Modifiers Associated with the FILE Keyword in ANIMATE for ECDA Data.....	15
Table 2.6 Modifiers Associated with the MEDIA Keyword in ANIMATE for ECDA Data.....	15
Table 2.7 Example Keyword File for ANIMATE for ECDA Data .....	16
Table 2.8 Modifiers Associated with the FILE Keyword in ANIMATE for ECEM Data .....	17
Table 2.9 Modifiers Associated with the SOILTYPE Keyword in ANIMATE for ECEM Data .....	18
Table 2.10 Modifiers Associated with the SOLUTION Keyword in ANIMATE for ECEM Data.....	18
Table 2.11 Example Keyword File for ANIMATE for ECEM Data .....	19
Table 2.12 Modifiers Associated with the FILE Keyword in ANIMATE for FCDA Data.....	20
Table 2.13 Modifiers Associated with the SOIL Keyword in ANIMATE for ECEM Data.....	21
Table 2.14 Example Keyword File for ANIMATE for FCDA Data .....	22
Table 2.15 Modifiers Associated with the ANALYTE Keyword in ANIMATE for HUMAN Data.....	23
Table 2.16 Modifiers Associated with the FILE Keyword in ANIMATE for HUMAN Data .....	23
Table 2.17 Modifiers Associated with the SOLUTION Keyword in ANIMATE for HUMAN Data.....	24
Table 2.18 Example Keyword File for ANIMATE for HUMAN Data .....	25
Table 3.1 Representative Coefficient Values for Background Concentrations in the Columbia River.....	27
Table 3.2 Excerpts from a biota.stoch file .....	28
Table 3.3 Example Keyword File for Back_MASS2.....	29
Table 3.4 Example Dissolved Concentration File for MASS2.....	30
Table 3.5 Example Particulate Concentration File for MASS2.....	30
Table 3.6 Example Sediment Loading File for MASS2 .....	30
Table 3.7 Modifiers Associated with the ANALYTE Keyword in Back_MASS2 .....	31
Table 4.1 Example Input File for CONSUME .....	36
Table 4.2 Excerpts from a CONSUME Report File .....	37
Table 5.1 Example Keyword File for the CRGRAB Code.....	40
Table 5.2 Annual River Release File fom CRGRAB .....	41
Table 5.3 Cumulative Release File from CRGRAB .....	42

Table 5.4 Total Release by Releasing Location from CRGRAB.....	42
Table 5.5 Modifiers Associated with the DEBUG Keyword in CRGRAB .....	43
Table 6.1 Example Keyword File for the CSCALE Code .....	50
Table 6.2 Modifiers Associated with the ANALYTE Keyword.....	52
Table 6.3 Modifiers Associated with the FILE Keyword .....	53
Table 7.1 Example Keyword File for ECDA.....	58
Table 7.2 Modifiers Associated with the ANALYTE Keyword for ECDA .....	60
Table 7.3 Modifiers Associated with the FILE Keyword for ECDA.....	62
Table 7.4 Modifiers Associated with the FILLECDA Keyword in the ESD File .....	64
Table 7.5 Modifiers Associated with the LOCATION Keyword in the ESD File .....	65
Table 8.1 Excerpts from an ECDA_ASCII Output File .....	70
Table 9.1 Example Keyword File for ECDA_BACK.....	74
Table 9.2 Modifiers Associated with the MEDIA Keyword in ANIMATE for ECDA Data.....	76
Table 10.1 Excerpts from an FCDA_ASCII Output File.....	80
Table 11.1 Example Keyword File for the FILLECDA Program.....	82
Table 11.2 Modifiers Associated with the DATA Keyword in FILLECDA .....	83
Table 12.1 Example Keyword File for the FILLFCDA Program.....	88
Table 12.2 Modifiers Associated with the DATA Keyword in FILLFCDA .....	89
Table 13.1 Example Keyword File for the GETTRACE Program .....	94
Table 13.2 Example Output File from the GETTRACE Code .....	94
Table 13.3 Modifiers Associated with the Analyte Type on the ANALYTE Keyword in GETTRACE ..	96
Table 13.4 Modifiers Associated with the Solution Type on the ANALYTE Keyword in GETTRACE ..	96
Table 14.1 Modifiers Associated with the EXTRACT keyword for GWGRAB.....	103
Table 14.2 Example Keyword File for the GWGRAB Program - Check Run Status .....	104
Table 14.3 Example Output File for the GWGRAB Program – Check Run Status .....	105
Table 14.4 Example Keyword File for the GWGRAB Program – Extract CFEST Results .....	105
Table 14.5 Example Output File for the GWGRAB Program – Extract CFEST Results.....	105
Table 15.1 Example Keyword File for HIGHDOSE .....	108
Table 15.2 Example Species Definition File for HIGHDOSE .....	108
Table 15.3 Example Results File from the HIGHDOSE Code.....	109
Table 15.4 Reformatted Output From the HIGHDOSE Code .....	109
Table 15.5 Modifiers Associated with the MEMORY Keyword in HIGHDOSE .....	111
Table 15.6 Modifiers Associated with the SOIL Keyword in HIGHDOSE .....	112
Table 15.7 Modifiers Associated with the SOLUTION Keyword in HIGHDOSE .....	113
Table 16.1 Example Keyword File for the HIGHIMPACT Program .....	116
Table 16.2 Example Results File from the HIGHIMPACT Code .....	116
Table 16.3 Modifiers Associated with the REALIZAT Keyword in HIGHIMPACT .....	118
Table 16.4 Modifiers Associated with the R_TYPE Keyword in HIGHIMPACT .....	119
Table 16.5 Modifiers Associated with the S_TYPE Keyword in HIGHIMPACT .....	120
Table 17.1 Example Keyword File for the HIGHMEDIA Program.....	124
Table 17.2 Example Results File from the HIGHMEDIA Code .....	125
Table 17.3 Modifiers Associated with the MEDIA Keyword in HIGHMEDIA .....	128
Table 17.4 Modifiers Associated with the REALIZAT Keyword in HIGHMEDIA.....	128
Table 18.1 Waste Streams Generated from Processing of Tank Wastes .....	131
Table 18.2 Structure of HTWOS_TDP.INP Control Data File .....	132
Table 18.3 Example HTWOS_TDP.INP File for HTWOS_TDP.....	133

Table 18.4 Structure of the HTWOS Annual Data File .....	134
Table 18.5 Excerpts from the HTWOS_TDP.CSV File written by HTWOS_TDP .....	135
Table 19.1 Example File HTWOS_LIST.INP for HTWOS_TLDP .....	138
Table 19.2 Excerpts from the HTWOS_LEAK.CSV File .....	139
Table 21.1 Example Keyword File for INGRAB .....	144
Table 22.1 Example Keyword File for INGRES .....	150
Table 22.2 Excerpts from an INGRES Results File.....	150
Table 23.1 Excerpts from a Keyword Control File for INPROC .....	157
Table 23.2 Aggregate Site Mapping File for INPROC.....	159
Table 23.3 Excerpted Records from an INPROC Inventory Action File.....	160
Table 23.4 Excerpts from a Disposal Action File Written by INPROC .....	161
Table 23.5 Excerpts from a Summary Action File Written by INPROC.....	161
Table 23.6 Excerpts from an Aggregate Keyword File Written by INPROC.....	162
Table 23.7 TYPE Modifiers Associated with the ANALYTE Keyword in INPROC .....	163
Table 23.8 Modifiers Associated with the FILE Keyword in INPROC .....	165
Table 23.9 Statistical Distributions on the STATRULE Keyword for INPROC.....	167
Table 25.1 Field Descriptions for Table Inventory_Data .....	175
Table 25.2 Field Descriptions for Table dbo_SITE.....	176
Table 25.3 Field Descriptions for Table dbo_Site-Subsite_List.....	177
Table 26.1 Example Keyword File for the INVEXT Code .....	182
Table 26.2 Subset of the Records in the INPROC-Generated Rule File.....	184
Table 26.3 Example Site Results File from the INVEXT Code .....	185
Table 26.4 Excerpts from a Tank Results File from the INVEXT Code.....	185
Table 26.5 Modifiers Associated with the FILE Keyword in the INVEXT Keyword File .....	187
Table 27.1 Example Keyword File for INVLOT.....	192
Table 28.1 Example Keyword File for the INVREPAR Code .....	200
Table 28.2 Modifiers Associated with the FILE Keyword in the INVREPAR Keyword File .....	202
Table 29.1 Example Keyword File for INVSUM.....	206
Table 29.2 Excerpted Records from an INVSUM Results File .....	207
Table 29.3 Excerpted Records from an INVSUM Summary Statistics File.....	208
Table 29.4 Modifiers Associated with the ANALYTE Keyword in INVSUM.....	209
Table 29.5 Modifiers Associated with the FILE Keyword in INVSUM .....	209
Table 31.1 Example Keyword File for MAKEU .....	216
Table 31.2 Excerpt from a MAKEU-generated Results File .....	216
Table 31.3 Modifiers Associated with the FILE Keyword in MAKEU Keyword File .....	217
Table 32.1 Example Keyword File for PICK_HYDRO .....	223
Table 32.2 Example Rock Basis Keyword File Output from the PICK_HYDRO Code.....	223
Table 32.3 Modifiers Associated with the BASIS Keyword in PICK_HYDRO .....	225
Table 32.4 Modifiers Associated with the FILE Keyword in PICK_HYDRO.....	226
Table 32.5 Modifiers Associated with the ITERATE Keyword in PICK_HYDRO .....	227
Table 32.6 Modifiers Associated with the METRIC_M Keyword in PICK_HYDRO .....	228
Table 32.7 Modifiers Associated with the METRIC_V Keyword in PICK_HYDRO .....	229
Table 32.8 Modifiers Associated with the RELAX Keyword in PICK_HYDRO.....	229
Table 32.9 Variable IDs Needed for the STOCHASTIC Keyword in PICK_HYDRO.....	231
Table 33.1 Parameters Associated with the Execution of the PRESTOMP Code.....	235
Table 33.2 Example STOMP Input and Output File “input” to the PRESTOMP Code.....	237

---

Table 35.1	Example Keyword File for RLGRAB .....	243
Table 37.1	Excerpted Lines from a SIMS Input Data File .....	253
Table 37.2	Excerpts from an Output File from SIMS.....	253
Table 38.1	Example Keyword File for STOCHASTIC .....	258
Table 38.2	Modifiers Associated with the DEBUG Keyword in STOCHASTIC .....	259
Table 38.3	Statistical Distributions Available in STOCHASTIC.....	262
Table 40.1	Example Keyword File for VOLEXT .....	268
Table 40.2	Excerpts from a VOLEXT–Generated Results File.....	269
Table 41.1	Example Keywords for a Single Release Site for the VZGRAB Program.....	274
Table 41.2	Example Keywords for Summing Over All Release Sites for the VZGRAB Program.....	274
Table 41.3	Example Keywords for Summing Over Selected Release Sites for the VZGRAB Program..	274
Table 41.4	Modifiers Associated with the EXTRACT Keyword in the VZGRAB Control File .....	276
Table 43.1	Contents of the Output File “watertable.idx” for the WTRTBL Program .....	282
Table 45.1	Common Statistical Distributions Available in SAC Codes.....	290





## 1.0 Introduction and Background

In 1999, the U.S. Department of Energy (DOE) initiated the development of an assessment tool that will enable users to model the movement of contaminants from all waste sites at Hanford through the vadose zone, groundwater, and Columbia River and to estimate the impact of contaminants on human health, ecology, and the local cultures and economy. This tool, named the System Assessment Capability (SAC), is an integrated system of computer models and databases used to assess the impact of waste remaining on the Hanford Site. The SAC will help decision makers and the public evaluate the cumulative effects of contamination from Hanford.

The design of the SAC resulted from extensive interactions with Hanford projects, regulators, Tribal Nations, and stakeholders. The approach taken in the assessment follows that advanced by regulatory agencies such as the U. S. Environmental Protection Agency (EPA) in its guidance on uncertainty analyses (Firestone et al. 1997) and ecological risk assessments (EPA 1998) and DOE in its radioactive waste management manual (DOE 1999a) and implementation guide (DOE 1999b) that support DOE Order 435.1 on radioactive waste management. The SAC also was designed with the intent to use it to perform the next composite analysis, an assessment first performed to satisfy Defense Nuclear Facility Safety Board recommendation 94-2. The approach taken is also consistent with the methods, characteristics, and controls associated with acceptable analyses as described by the Columbia River Comprehensive Impact Assessment (CRCIA) team (DOE 1998).

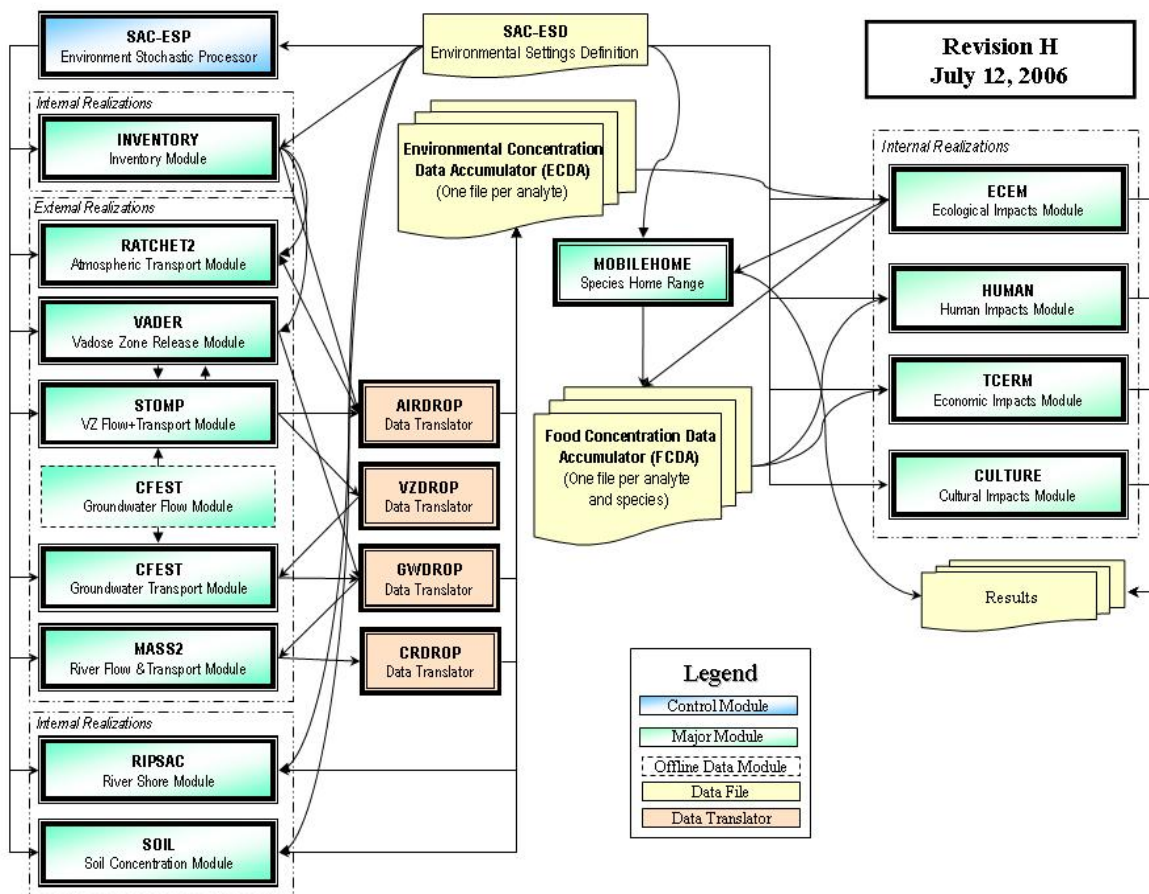
### 1.1 Overview of the SAC Systems Code

The SAC Systems Code is a tool used to simulate the migration of contaminants (analytes) present on the Hanford Site and assess the potential impacts of the analytes, including dose to humans, socio-cultural impacts, economic impacts, and ecological impacts. The system of codes includes computer programs, electronic data libraries, and data formatting processors (or data translators). The relationships among code modules that make up the SAC Systems Code are illustrated in Figure 1.1.

Major modules appearing on the left side of the diagram perform inventory and transport calculations, providing estimates of the concentrations of analytes in various media. Modules shown on the right perform calculations related to impacts from the contaminated media. Impacts include potential effects on humans, the ecology of the area, the economy of the region, and the proximity of contaminants to social and cultural resources.

The general approach to handling uncertainty in SAC, Rev. 1, is a Monte Carlo approach. Conceptually, a value is generated for every stochastic parameter in the code (the entire sequence of modules from inventory through transport and impacts), and then the simulation is executed to obtain an output value or result. This process is often called one *realization*. The entire process is then repeated, obtaining another result that is different from the first but as equally likely to occur as the first result. After repeating this process a number of times, one has a set of equally likely consequences that represent the statistical distribution of all outcomes. Several specialized sampling techniques have been developed to reduce the number of realizations required in a Monte Carlo analysis to obtain a satisfactory description of the output distribution. One of the techniques, called Latin Hypercube Sampling (Iman and Conover 1982), has proven successful for mass transport applications in groundwater systems. The general Monte Carlo approach still applies, and the specific values of the input parameters are chosen from the same statistical

distributions; however, the sampling scheme spreads the values in a way that reduces sampling variability while also supporting a correlation structure between input variables.



**Figure 1.1** Module Information Flow for the SAC Rev. 1 Systems Code

## 1.2 Purpose of This Document

This document contains detailed user instructions for a suite of utility codes developed for Rev. 1 of the SAC. The suite of computer codes for Rev. 1 of SAC performs many functions. User instructions for the main transport and impacts codes are provided in Eslinger et al. (2006a, 2006b). It also performs inventory tracking, release of contaminants to the environment, and transport of contaminants through the unsaturated zone, saturated zone, and in the river. User instructions and supporting theoretical information for the codes on the left side of Figure 1.1 are provided in the following documents:

- Air transport model - RATCHET2 (Ramsdell and Rishel 2006)
- Vadose zone flow and transport model - STOMP (White and Oostrom 2000)
- Groundwater flow and transport model - CFEST (Freedman et al. 2006)
- River flow and transport model - MASS2 (Perkins and Richmond 2004a, 2004b)
- All SAC-specific transport modules (Eslinger et al. 2006a)

User instructions and supporting theoretical information for the codes on the right side of Figure 1.1 are provided in the following document:

- All SAC-specific impact modules (Eslinger et al. 2006b)

### 1.3 Overview of the Utility Codes

An overview of the utility codes described in this document is provided in Table 1.1. The utility codes are grouped by functional area shown in Figure 1.1. Some of the utility codes, such as INPROC, form an integral part of the inventory, release, transport, and impact generation functions. Other utility codes, such as VZGRAB, are used to summarize and extract data generated by the primary codes. Some codes such as MAKEU or CSCALE efficiently create data that could have been generated by the main sequence of codes.

**Table 1.1** Overview of Utility Codes

Code Name	Purpose
<b>Inventory-Related Functions</b>	
HTWOS-TDP	This program reformats and decay corrects HTWOS (Hanford Tank Waste Operations Simulator) data so it can be imported into the inventory database.
HTWOS-TDLP	This program extracts and reformats tank leak data from the HTWOS (Hanford Tank Waste Operations Simulator) data so it can be imported into the inventory database.
INGRAB	This program summarizes the analyte inventory and waste volumes in the inventory data generated by the INVENTORY code. Data extracted (from the “inventory.all” file) may be annual or accumulated annual values, and radioactive analyte totals are decay corrected.
INGRES	This program summarizes the analyte inventory and waste volumes in the inventory data generated by the INVENTORY code. Data extracted (from a single “inv*.res” file) may be annual or accumulated annual values, and radioactive analyte totals are decay corrected.
INPROC	This program reads output from the inventory database, applies statistical and data fill-in rules, and generates a stochastic disposal action file for use in the INVENTORY code. It also generates WASTEMAP keywords for the INVENTORY code.
Inventory Database	This Microsoft Access-based database compiles inventory data from all sources. This database provides inputs for the INPROC code to start the inventory processing for a simulation case.
INVEXT	This program allows the user to collect and summarize information in the inventory generated by the INVENTORY code. The most commonly used output is a table indicating inventory for each analyte subdivided by data type, where data type is record data, surrogate rule application, SIM data, or HTWOS data. This program also extracts data for the past leaks, future leaks, and residuals for each Hanford waste tank.
INVLOT	This program generates data files of inventory on an annual time step for user-specified combinations of analytes, waste sites, and waste locations that can be used in a plotting package.
INVREPAR	This program allows the user to split the inventory at one site into multiple sites. The nominal use of this code is to split the inventory for B Pond liquid wastes into a main disposal and three additional lobes for a user-specified range of years.

Code Name	Purpose
INVSUM	This program generates summary information for inventory data. It can be used to identify sites for which no inventory was generated.
MAKEU	This program reads a single inventory results (“inv*.res”) file and writes a modified “inv*.res” file that contains the element uranium computed as the sum of all the uranium isotope values.
SIMS	This program reformats and decay corrects SIM (Soil Inventory Model) data so they can be imported into the inventory database.
VOEXT	This program extracts a sum of volumes by waste form for selected waste sites and selected years from the results files built by the INVENTORY code.
<b>Release and Vadose Zone-Related Functions</b>	
GWGRAB	This program checks on the status of groundwater transport (CFEST) runs and also can extract summary results from CFEST runs once they have completed.
L3IDIFF	This program differences nodal fluid sources in two otherwise identical CFEST .l3i input files. This utility is expressly designed to support the calculation of artificial fluid discharges.
PICK_HYDRO	This program defines constrained sampling for the parameters in the vadose zone flow model (STOMP code). This logic is invoked only when random sampling from the statistical distributions for the parameters result in unreasonable flow or transport results due to unrealistic combinations of the input parameters.
PRESTOMP	This program is a preprocessor utility code for STOMP that refines a STOMP input file before solving for coupled fluid flow and analyte transport.
RELUP	This program is a post-processor utility code for STOMP that provides the means to restrict the minimum time duration for a release record written by STOMP.
RLGRAB	This program consolidates waste release profiles generated by VADER for a given scenario as specified by one or more Sites, a specific analyte, and a set of waste release model parameters, over N realizations.
SETRES	This program is a small processing utility code used to rename files following a STOMP flow simulation to prepare for a STOMP transport simulation restart in the same directory.
VZGRAB	This program extracts and consolidates data for several variables used in the vadose zone flow and transport calculations.
VZSWAP	This program provides a means to swap third-party vadose zones release results into or out of the SAC system.
WTRBL and RDWTRTBL	These programs are utility codes that read CFEST binary files for a completed transient flow solution and write a pair of files that ESP uses for fast access to the water table elevation at CFEST nodes nearest each vadose zone site when preparing STOMP input files.
<b>Saturated Zone-Related Functions</b>	
<b>River-Related Functions</b>	
BACK_MASS2	This program generates files containing stochastic representation of background water and sediment concentrations that will be used in the river transport code MASS2.
CRGRAB	This program allows the user to collect and summarize the analyte mass or activity to the Columbia River as prepared for use by MASS2.

Code Name	Purpose
<b>Median Value Generation-Related Functions</b>	
Imp_Med	These programs convert a stochastic keyword file into a median-values keyword file. This process applies to keyword files for the ECEM, HUMAN, and TCERM codes.
Inv_Med	These programs convert a stochastic disposal action file into a median-values disposal action file.
Sto_Med	These programs convert a stochastic keyword file containing data for STOMP (actually used in the ESPcode) into a median-values keyword file.
STOCHASTIC	This program can generate sample data sets and summary statistics for any stochastic variable.
<b>ECDA File-Related Functions</b>	
CSCALE	This program reads one or more environmental concentration data accumulator (ECDA) files and writes another file that is a linear combination of the input files. The linear combination weights can be a function of time. This program can be used to build a file of decay product concentrations given a parent file.
ECDA	This program creates the binary environmental concentration data accumulator (ECDA) files and initializes the contents for each medium.
ECDA_ASCII	This program converts a user-specified number of records in an ECDA file from binary storage mode to ASCII storage mode to allow visual inspection.
ECDA_BACK	This program reads an ECDA file, identifies a target set of data for each physical location, and writes the target data to the same physical location for all succeeding time steps to fill in background data for later times for an ECDA file that was only partially filled from a run of the river code (MASS2).
FCDA_ASCII	This program converts a user-specified number of records in an FCDA (food concentration data) file from binary storage mode to ASCII storage mode to allow visual inspection.
FILLECDA	This program allows user-specified values of concentration data to be inserted into an existing ECDA file.
FILLFCDA	This program allows user-specified values of concentration data to be inserted into an existing FCDA file.
HIGHMEDIA	This program allows extraction of the largest media value in an ECDA file for a suite of user-specified locations and years.
<b>Impacts-Related Functions</b>	
ANIMATE	This program extracts data from the ECDA or outputs from ECEM or HUMAN and prepares data files for performing animations in the Tecplot <sup>®</sup> plotting program.
CONSUME	This program reads a predation matrix and converts it into CONSUME keywords for use in the ecological impacts model (ECEM).
GETTRACE	This program gathers up a suite of results by analyte after the HIGHIMPACT code has identified the location by year where the maximum dose occurs (deterministic runs only).
HIGHDOSE	This program generates a summary table of the ecological species with the highest impacts for a suite of user-specified solutions.
HIGHIMPACT	This program extracts the highest impact (from a suite of possible impacts) for each year for a user-specified set of impact locations.

© The Tecplot software is copyrighted by Tecplot, Inc., Bellevue, WA

## 1.4 General Rules for Reading Keyword Descriptions

Many of the programs in this document are controlled through the use of data files containing text entries called keyword records. A keyword record is typically constructed from an identifying “keyword” name (such as “ANALYTE”) and data associated with the keyword (such as the names of analytes used in a simulation run). A description of the general syntax for keyword language is provided in Section 44.0. In the keyword descriptions of this document, some data are optional for a particular problem definition and some are required. For simplicity of interpretation, the following additional rules apply to the *keyword descriptions* that you will see in this document:

- Data that are required are enclosed in square brackets. For example, if AB were required, it would be denoted by [AB].
- If only one of the three items AB, BC, CD were required, it would be written as [AB|BC|CD]. The vertical bars indicate that the user must select one of the items in the list.
- Optional items are enclosed in normal brackets. For example, if DE were an optional entry, it would be denoted by {DE}.
- The {} or [] brackets or | vertical bars are NOT entered when constructing the actual keyword record: the keyword syntax specifications in this document use them only to indicate whether an item for the keyword is {optional} or [required] or a [required|choice].
- The keyword name can contain one or more characters; however, only the first eight characters are used (for example, REALIZAT has the same effect as REALIZATION).
- In some instances, numerical values or quote strings are associated with a modifier. This association is indicated by using the equal ( = ) symbol. The = symbol is not required but may be used when the keyword is constructed. When a numerical value or quote string is associated with a modifier, it must be entered on the input line directly after the modifier. Quote strings must be enclosed in double quotation marks. For example:

```
FILE C_ECDA ANALYTE="U" NAME="/home/CPO/ecda/U_CPO.dat" CREATE
```

- Although keyword examples are indented in this document for the purpose of display, actual keyword records always start in column 1. Continuation records, which are treated as a continuation of a keyword record, appear further indented in the examples. For example:

```
SITES "300_RLWS" "300_RRLWS" "300_VTS" "300-121"  
      "300-123" "300-16" "300-2" "300-214" "300-224"
```

Note that the term “keyword” in SAC documentation often refers to the entire keyword record: the keyword name plus any associated data items that follow.

## 2.0 ANIMATE – Animation of Impact Code Results

### 2.1 Overview

ANIMATE is a utility code that extracts data and sets up input files for Tecplot for the purpose of making time-series animations. Data are handled from five separate parts of the Systems Assessment Capability, (SAC), Rev. 1. The data types are shown in Table 2.1:

**Table 2.1** Data Types Used in the ANIMATE Utility Code

Data Type	Description
CULTURE	Data written to the “MAP” file by the CULTURE code
ECDA	Media concentrations in the binary ECDA files
ECM	Any data written to the “DETAILS” file by the ECM code
FCDA	Food concentrations in the binary FCDA files
HUMAN	Any data written to the “DETAILS” file by the HUMAN code

#### 2.1.1 Location in the Processing Sequence

The ANIMATE program can be used anytime after one or more of the following:

- The CULTURE program has created and written any data to the CULTURE “MAP” file.
- The ECDA program has created media concentrations in the binary ECDA files.
- The ECM program has created and written any data to the ECM “DETAILS” file.
- The ECM program has created food concentrations in the binary FCDA files.
- The HUMAN program has created and written any data to the HUMAN “DETAILS” file.

#### 2.1.2 How the Code Is Invoked

ANIMATE can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), ANIMATE executes in a DOS box. A run of ANIMATE is initiated by entering the following command line:

```
ANIMATE "Keyfilename"
```

Under the Linux operating system ANIMATE is executed through any of the following Bourne Shell or C Shell commands:

```
animate-1.exe "Keyfilename"
```

For these commands, “ANIMATE.EXE” or “animate-1.exe” is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If ANIMATE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If ANIMATE cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

### 2.1.3 Memory Requirements

The ANIMATE program has moderate memory requirements. The major use of memory is storing the information from all DATA keywords before any is processed. If memory requirements become excessive, the problem can be subdivided into multiple runs, each using a subset of the data.

## 2.2 File Definitions

The ANIMATE program reads up to five input files and writes to two or more output files. These files are described in the following paragraphs.

### 2.2.1 Input Files

#### 2.2.1.1 ANIMATE Keyword File

The ANIMATE keyword file contains control information. Example files are provided in each of the sections describing the keyword input for the particular data types. Detailed definitions of the keywords are provided in Sections 2.3 through 2.8.

#### 2.2.1.2 Other Input Files

In addition to the keyword input file, the ANIMATE program also reads several additional input files. The files needed are based on the category selected for the animation. Only one category can be selected for each run of the ANIMATE code. The following are the files needed to run the ANIMATE code based on the type of animation selected:

##### **CULTURE MAP:**

- CULTURE header file
- Map data file
- Coordinates file, this file defines location (ID,X,Y)

##### **ECDA UPLAND:**

- ECDA concentration file
- ECDA map index file
- Connectivity files can be for all locations (2828) and for Outside\_core locations (2438)
- Coordinates file, this file defines location (ID,X,Y)

##### **HUMAN UPLAND:**

- HUMAN header file
- Risk detailed data file
- Connectivity files can be for all locations (2828) and for Outside\_core locations (2438)
- Coordinates file, this file defines location (ID,X,Y)

##### **FOOD UPLAND:**

- FCDA concentration file
- FCDA map index file



- Connectivity files can be for all locations (2828) and for Outside\_core locations (2438)
- Coordinates file, this file defines location (ID,X,Y)

**ECEM UPLAND:**

- ECEM header file
- Risk detailed data file
- Connectivity files can be for all locations (2828) and for Outside\_core locations (2438)
- Coordinates file, this file defines location (ID,X,Y)

**ECDA RIPARIAN/AQUATIC** case then these files are:

- ECDA concentration file
- ECDA map index file
- Coordinates file, this file defines location (ID,X,Y)

**HUMAN RIPARIAN/AQUATIC** case and these files are:

- Header file
- Risk detailed data file
- Coordinates file, this file defines location (ID,X,Y)

**FOOD RIPARIAN/AQUATIC** case and these files are:

- FCDA concentration file
- FCDA map index file
- Coordinates file, this file defines location (ID,X,Y)

**ECEM RIPARIAN/AQUATIC** case and these files are:

- Header file
- Risk detailed data file
- Coordinates file, this file defines location (ID,X,Y)

## **2.2.2 Output Files**

The code writes to two output files. These files are a report file and a Tecplot visualization file. The Tecplot file defines years, connectivity, and concentrations for display.

### **2.2.2.1 Report File**

The report file contains summary information from the run of the ANIMATE code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the ANIMATE code.

### **2.2.2.2 Tecplot File**

The Tecplot file contains the data that will be read by Tecplot to produce the animation. The file contains the coordinate information and the selected data for the animation.

## 2.3 Top-Level Keywords for Every Run of ANIMATE

In general, the keywords for ANIMATE can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 2.3.1 CATEGORY Keyword for ANIMATE

The CATEGORY keyword (mandatory) defines the data category to be processed. The types of data that can be processed are defined in Table 2.2. The following is this keyword's syntax:

```
CATEGORY [MODIFIER]
```

**Table 2.2** Modifiers Associated with the CATEGORY Keyword in ANIMATE

Modifier	Description
CULTURE	Animate a map file generated through use of the MAP keyword
ECDA	Animate environmental media concentration data (Environmental Concentration Data Accumulator – ECDA).
ECM	Animate any of the solutions available through use of the DETAILS keyword
FCDA	Animate food concentrations (Food Concentration Data Accumulator – FCDA)
HUMAN	Animate any of the solutions available through use of the DETAILS keyword

An example of this keyword that uses the environmental media concentration data:

```
CATEGORY ECDA
```

### 2.3.2 DEBUG Keyword for ANIMATE

The DEBUG keyword is used to activate the writing of intermediate information to the report file. The following is this keyword's syntax:

```
DEBUG [CULTURE | ECDA | ECM | FCDA | HUMAN]
```

### 2.3.3 END Keyword for ANIMATE

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 2.3.4 LEGEND Keyword for ANIMATE

The LEGEND keyword is used to enter the names of the plot title and the header for the contour legend. The following is this keyword's syntax:

```
LEGEND [TITLE= "quote 1"] [CONTOUR="quote 2"]
```

The quote string associated with the TITLE modifier is used to enter a one-line plot title. The quote string associated with the CONTOUR modifier is used to enter a one-phrase label for the contour definitions. Some examples of the LEGEND keyword are as follows:

```
LEGEND TITLE = "Terrestrial Plant" CONTOUR = "Concentration"  
LEGEND TITLE = "H3 UpDrink" CONTOUR = "Dose (mrem)"
```

### 2.3.5 LOCATION Keyword for ANIMATE

The LOCATION keyword identifies the location type that will be used in the animation. All locations with the selected location type are used. The following is this keyword's syntax:

```
LOCATION [UPLAND | RIPARIAN | AQUATIC]
```

An example of this keyword that uses irrigated soil with groundwater:

```
LOCATION UPLAND
```

### 2.3.6 PICK\_REL Keyword for ANIMATE

The PICK\_REL keyword defines the realization to process. If the CULTURE modifier is entered on the CATEGORY keyword, then PICK\_REL must be 1. The following is this keyword's syntax:

```
PICK_REL value1
```

The following keyword record chooses realization number 10:

```
PICK_REL 10
```

### 2.3.7 REPORT Keyword for ANIMATE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/SystemCodes/Cultural/Test1.rpt"
```

### 2.3.8 TIMES Keyword for ANIMATE

The TIMES keyword identifies the times (years) at which the calculations are to be performed for a single case. The following is this keyword's syntax:

```
TIMES [ALL | LIST [T1] {T2} ... {Tn}]
```

All times in the ESD keyword file are used when the modifier ALL is present. A list of specific times can be defined by entering the modifier LIST and a set of numerical values identifying the years of interest. The specific years entered must be defined in the ESD keyword file. All times are included in the first example shown below. The second example defines a list of six years.

```
TIMES ALL  
TIMES LIST 2020 2075 3014 3050 4000 12050
```

### 2.3.9 TITLE Keyword for ANIMATE

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the ANIMATE code."
```

### 2.3.10 USER Keyword for ANIMATE

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 2.3.11 VERBOSE Keyword for ANIMATE

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

## 2.4 Keywords and Keyword File Specific to CULTURE Data

The keywords needed for an animation of CULTURE map data are given in this section. These keywords are needed in addition to the keywords described in Section 2.3.

### 2.4.1 FILE Keyword

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"} {modifier3="quote 3"}  
{modifier4="quote 4"} {modifier5="quote 5"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 2.3.

**Table 2.3** Modifiers Associated with the FILE Keyword in ANIMATE for CULTURE Data

Modifier	Description
CONNECT	Tecplot connectivity file (needed for plot of upland locations)
COORDS	Coordinate data (This defines location (ID,X,Y))
HEADER	Header file created by the CULTURE code
MAP	Map data
RESULTS	Output file used for Tecplot visualization

### 2.4.2 MAP Keyword

The MAP keyword is used to define the map for which data will be processed. The following is this keyword's syntax

```
MAP [ "quote" ]
```

The single quote string must be a single map ID from the set of the maps generated by a run of the CULTURE code. An example of this keyword is the following:

```
MAP "Map_U"
```

### 2.4.3 Example Keyword File for CULTURE Map Data

An example keyword file for running an animation for the CULTURE map data is shown in Table 2.4.

**Table 2.4** Example Keyword File for ANIMATE for CULTURE Upland Data

```
REPORT "Test_Animate_CULTURE.rpt"
TITLE "Animation Test for HUMAN UPLAND locations"
USER "Carmen Arimescu"
OS WINDOWS !UNIX ! This matches information in the header file
VERBOSE
LOCATION UPLAND ! RIPARIAN or AQUATIC
!-----
CATEGORY CULTURE !---> Only one category of data allowed per run of the code
FILE HEADER "U_History2_Culture_DW_map10%.hdr" ! Header file
FILE MAP "U_History2_Culture_DW_map10%.csv" ! Risk detailed data
! Files needed
FILE RESULTS "U_History2_Culture_DW_map10%.out.dat" !Output file used for
tecplot visualization
! Connectivity files can be for all locations(2828) and for Outside_core
locations(2438)
FILE CONNECT "Core_Connectivity.txt" ! TecPlot connectivity file
FILE COORDIN "Core_Loc.csv" ! Coordinates for the connectivity!
! Pick one or more times
TIMES ALL
! Pick a single Map
MAP "Map_U"
! Pick a single data value to analyte
PICK_REL 1
LEGEND TITLE = "U UpDrink" CONTOUR = "Percent"
END
```

## 2.5 Keywords and Keyword File Specific to ECDA Data

The keywords needed for an animation of ECDA concentration data are given in this section. These keywords are needed in addition to the keywords described in Section 2.3.

### 2.5.1 ANALYTE Keyword

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax

```
ANALYTE [ "quote" ]
```

The single quote string must be a single analyte ID from the set of the analytes in the ESD keyword file. An example of this keyword is the following:

```
ANALYTE "C14"
```

### 2.5.2 FILE Keyword

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"} {modifier3="quote 3"}
{modifier4="quote 4"} {modifier5="quote 5"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 2.5.

**Table 2.5** Modifiers Associated with the FILE Keyword in ANIMATE for ECDA Data

Modifier	Description
CONNECT	Tecplot connectivity file (not needed with river line plot)
COORDS	Coordinate data (This defines location (ID,X,Y))
C_ECDA	ECDA concentration file
I_ECDA	ECDA map index file
RESULTS	Output file used for Tecplot visualization

### 2.5.3 MEDIA Keyword

The MEDIA keyword is used to define the medium for which data will be processed for a single case. The following is this keyword's syntax:

```
MEDIA [ "quote 1" ]
```

The single quote string is case sensitive, four characters in length. The modifiers associated with the FILE keyword are given in Table 2.6.

**Table 2.6** Modifiers Associated with the MEDIA Keyword in ANIMATE for ECDA Data

Modifier	Description
GWAT	concentrations in groundwater ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEEP	concentrations in seep water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SWAT	concentrations in surface water (river) ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
PWAT	concentrations in river bottom pore water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEDI	concentrations in river bottom sediment ( $\text{Ci}/\text{kg}_{\text{sediment}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{sediment}}$ )
SORP	concentrations in riparian zone soil (land surface) ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SODR	concentrations in upland soil (land surface) with no irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOGW	concentrations in upland soil (land surface) with groundwater irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOSW	concentrations in upland soil (land surface) with surface water irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
AIRC	concentrations in air ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
AIRD	air deposition rates ( $\text{Ci}/\text{m}^2/\text{yr}$ or $\text{kg}/\text{m}^2/\text{yr}$ )

An example of this keyword that uses concentrations in upland soil with surface water irrigation is the following:

```
MEDIA "GWAT"
```

## 2.5.4 UNITS Keyword

The UNITS keyword is used to define the output units for the animation. The following is this keyword's syntax:

```
UNITS OUTPUT "quote1" FACTOR value1
```

The quote string is the desired output units, and the value is the units conversion factor to translate from the units output by the original code to the desired output units for the animation. An example of this keyword is the following:

```
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9 !Convert Ci/m^3 to pCi/L
```

## 2.5.5 Example Keyword File for ECDA Data

An example keyword file for running an animation for ECDA concentration data is shown in Table 2.7.

**Table 2.7** Example Keyword File for ANIMATE for ECDA Data

```
REPORT "Test_Animate_ECDA.rpt"
TITLE "Animation for ECDA Upland locations"
USER "Carmen Arimescu"
OS WINDOWS !UNIX ! This matches information in the SacView header file
VERBOSE
LOCATION UPLAND !RIPARIAN or AQUATIC
!-----
CATEGORY ECDA !---> Only one category of data allowed per run of the code
FILE C_ECDA "CA_Ref_A_C14.bin" ! Name of ECDA concentration file
FILE I_ECDA "CA_Ref_A_ECDA.idx" ! Name of ECDA map index file
DEBUG ECDA
! Files needed
FILE RESULTS "C14_Ecda_out.dat" !Output file
FILE CONNECT "All_connectivity.txt" ! TecPlot connectivity file
FILE COORDIN "All_locations.csv" ! Coordinates for the connectivity
! Pick one or more times
TIMES ALL
! Pick a single analyte
ANALYTE "C14"
! Pick a medium
MEDIA "GWAT"
! Pick a single realization animate
PICK_REL 1
! Set the output units and legend labels
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9 !Convert Ci/m^3 to pCi/L
LEGEND TITLE = "Animation for ECDA Upland locations" CONTOUR =
"Concentration"
!
END
```



## 2.6 Keywords and Keyword File Specific to ECEM Data

The keywords needed for an animation of ECEM detailed data are given in this section. These keywords are needed in addition to the keywords described in Section 2.3.

### 2.6.1 ANALYTE Keyword

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax

```
ANALYTE [ "quote" ]
```

The single quote string must be a single analyte ID from the set of the analytes in the ESD keyword file. An example of this keyword is the following:

```
ANALYTE "C14"
```

### 2.6.2 FILE Keyword

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1" ] {modifier2="quote 2" } {modifier3="quote 3" }  
{modifier4="quote 4" } {modifier5="quote 5" }
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 2.8.

**Table 2.8** Modifiers Associated with the FILE Keyword in ANIMATE for ECEM Data

Modifier	Description
CONNECT	Tecplot connectivity file (not needed with river line plot)
COORDS	Coordinate data (This defines location (ID,X,Y))
HEADER	Header file
DETAIL	Detailed data
RESULTS	Output file used for Tecplot visualization

### 2.6.3 SOILTYPE Keyword

The SOILTYPE keyword is used to define the solution type for which data will be processed when the CATEGORY is ECEM. The following is this keyword's syntax:

```
SOILTYPE [ "quote 1" ]
```

The single quote string is case sensitive and six characters in length. The modifiers associated with the SOILTYPE keyword are given in Table 2.9.

**Table 2.9** Modifiers Associated with the SOILTYPE Keyword in ANIMATE for ECEM Data

Modifier	Description
NONE	Applies to all aquatic locations
SORP	Riparian locations
SODR	Dry (non irrigated) upland locations
SOGW	Groundwater irrigated upland locations
SOSW	Surface water irrigated upland locations

An example of this keyword is the following:

```
SOILTYPE "SOGW"
```

## 2.6.4 SOLUTION Keyword

The SOLUTION is used to define the solution type for which data will be processed. The following is this keyword's syntax:

```
SOLUTION [ "quote 1" ]
```

The single quote string is case sensitive, six characters in length. For the CATEGORY ECEM, the modifiers associated with the SOILTYPE keyword are given in Table 2.10.

**Table 2.10** Modifiers Associated with the SOLUTION Keyword in ANIMATE for ECEM Data

Modifier	Description
BURDEN	Body burdens (pCi or µg/kg wet wt)
DOSRAD	Radioactive dose (rem)
BMTISS	Ratios to tissue benchmarks (unitless)
SUMRAD	Sum of radioactive dose (rem)
CONCEN	Media concentrations (see MEDIA keyword modifiers for units)

An example of this keyword is the following:

```
SOLUTION "DOSRAD"
```

## 2.6.5 SPECIES Keyword

The SPECIES keyword is used to define the species for which data will be processed. The following is this keyword's syntax

```
SPECIES [ "quote" ]
```

The single quote string must be a single SPECIES ID from the set of the SPECIES in the ESD keyword file. Example of this keyword :

```
SPECIES "RMALRD"
```

## 2.6.6 UNITS Keyword

The UNITS keyword is used to define the output units for the animation. The following is this keyword's syntax:

```
UNITS OUTPUT "quotel" FACTOR value1
```

The quote string is the desired output units, and the value is the units conversion factor to translate from the units output by the original code to the desired output units for the animation. An example of this keyword is the following:

```
UNITS OUTPUT="rad/d" FACTOR = 1.0 !No conversion needed
```

## 2.6.7 Example Keyword File for ECEM Data

An example keyword file for running an animation for ECEM detailed data is shown in Table 2.11.

**Table 2.11** Example Keyword File for ANIMATE for ECEM Data

```
REPORT "Test_Animate_ECEM.rpt"
TITLE "Animation Test for ECEM RIPARIAN locations"
USER "Carmen Arimescu"
OS WINDOWS !UNIX ! This matches information in the header file
VERBOSE
LOCATION RIPARIAN ! UPLAND or AQUATIC
!-----
CATEGORY ECEM !---> Only one category of data allowed per run of the code
FILE HEADER "Ecem_CA_Ref_A_foods.hdr" ! Header file for SACVIEW
FILE DETAIL "Ecem_CA_Ref_A_det_foods.csv" ! Risk detailed data
DEBUG ECEM
! Files needed
FILE RESULTS "ECEM_RP_out.dat" !Output file
FILE CONNECT "All_connectivity.txt" ! TecPlot connectivity file
FILE COORDIN "rip_animation.csv" ! Coordinates for the connectivity
! Pick one or more times
TIMES LIST 2000 2005
! Pick a soil type
SOILTYPE "SORP"
! Pick a single analyte
ANALYTE "H3"
! Pick a species
SPECIES "RMLBRY"
! Pick a solution
SOLUTION "DOSRAD"
! Pick a single realization to animate
PICK_REL 1
! Set the output units
UNITS OUTPUT="rad/day" FACTOR = 1.0 !No units changes
LEGEND TITLE = "none" CONTOUR = "none"
!
END
```

## 2.7 Keywords and Keyword File Specific to FCDA Data

The keywords needed for an animation of FCDA food concentration data are given in this section. These keywords are needed in addition to the keyword described in Section 2.3.

### 2.7.1 ANALYTE Keyword

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax

```
ANALYTE [ "quote" ]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string “-Rads-” if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANALYTE "C14"  
ANALYTE "-Rads-"
```

### 2.7.2 FILE Keyword

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1" ] {modifier2="quote 2" } {modifier3="quote 3" }  
{modifier4="quote 4" } {modifier5="quote 5" }
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 2.12.

**Table 2.12** Modifiers Associated with the FILE Keyword in ANIMATE for FCDA Data

Modifier	Description
CONNECT	Tecplot connectivity file (not needed with river line plot)
COORDS	Coordinate data (This defines location (ID,X,Y))
C FCDA	FCDA concentration file
I FCDA	FCDA map index file
RESULTS	Output file used for Tecplot visualization

### 2.7.3 SOIL Keyword

The SOIL keyword is used to the solution type for which data will be processed when the CATEGORY is FCDA. The following is this keyword's syntax:

```
SOIL [ "quote 1" ]
```

The single quote string is case sensitive and is six characters in length. The modifiers associated with the SOIL keyword are given in Table 2.13.

**Table 2.13** Modifiers Associated with the SOIL Keyword in ANIMATE for ECEM Data

Modifier	Description
SODR	Dry (non irrigated) upland locations
SOGW	Groundwater irrigated upland locations
SOSW	Surface water irrigated upland locations

An example of this keyword is the following:

```
SOIL "SOGW"
```

### 2.7.4 SPECIES Keyword

The SPECIES keyword is used to define the species for which data will be processed. The following is this keyword's syntax

```
SPECIES ["quote"]
```

The single quote string must be a single SPECIES ID from the set of the SPECIES in the ESD keyword file. Example of this keyword :

```
SPECIES "RMALRD"
```

### 2.7.5 UNITS Keyword

The UNITS keyword is used to define the output units for the animation. The following is this keyword's syntax:

```
UNITS OUTPUT "quotel" FACTOR value1
```

The quote string is the desired output units, and the value is the units conversion factor to translate from the units output by the original code to the desired output units for the animation. An example of this keyword is the following:

```
UNITS OUTPUT="pCi/kg" FACTOR = 1.0 !No conversion needed
```

### 2.7.6 Example Keyword File for FCDA Data

An example keyword file for running an animation for FCDA food concentration data is shown in Table 2.14.

**Table 2.14** Example Keyword File for ANIMATE for FCDA Data

```
REPORT "Test_Animate_FCDA.rpt"
TITLE "Animation Test for FCDA Upland locations"
USER "Carmen Arimescu"
OS WINDOWS !UNIX ! This matches information in the SacView header file
VERBOSE
LOCATION UPLAND !RIPARIAN or AQUATIC
!-----
-----
CATEGORY FCDA !---> Only one category of data allowed per run of the code
FILE C_FCDA "Food_H3_ULFVEG.fod" ! Name of FCDA concentration file
FILE I_FCDA "Ecem_CA_Ref_A_map_foods.dat" ! Name of FCDA map index file
DEBUG FCDA
! Files needed
FILE RESULTS "H3_ULFVEG_out.dat" !Output file
FILE CONNECT "All_connectivity.txt" ! TecPlot connectivity file
FILE COORDIN "All_locations.csv" ! Coordinates for the connectivity
! Pick one or more times
TIMES LIST 1945 1990 1995
! Pick a single analyte
ANALYTE "H3"
! Pick a soil type
SOIL "SOGW" !"SODR" SOSW"
! Pick a species
SPECIES "ULFVEG"
! Pick a single realization to animate
PICK_REL 1
! Set the output units
UNITS OUTPUT="pCi/kg" FACTOR = 1.0 !No conversion needed
LEGEND TITLE = "Terrestrial Plant" CONTOUR = "Concentration"
!
END
```

## 2.8 Keywords and Keyword File Specific to HUMAN Data

The keywords needed for an animation of HUMAN detailed data are given in this section. These keywords are needed in addition to the keywords described in Section 2.3.

### 2.8.1 ANALTYPE Keyword

The ANALTYPE keyword is used to define the analyte type for which data will be processed. The following is this keyword's syntax

```
ANALTYPE [ "quote" ]
```

The single quote string is case sensitive. The modifiers associated with the ANALTYPE keyword are given in Table 2.15.

**Table 2.15** Modifiers Associated with the ANALYTE Keyword in ANIMATE for HUMAN Data

Modifier	Description
ALL	The analyte field is used for a quantity that is a sum over analytes
CAR	The analyte is a carcinogenic chemical
CONMEDIA	The analyte field is used to select an environmental medium
CONFOODS	The analyte field is used to select a food species
HAZ	The analyte is a hazardous chemical
RAD	The analyte is a radionuclide

Example of this keyword :

```
ANALYTE "Rad"
```

### 2.8.2 ANALYTE Keyword

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax

```
ANALYTE ["quote"]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string "-Rads-" if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANALYTE "C14"
ANALYTE "-Rads-"
```

### 2.8.3 FILE Keyword

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"} {modifier3="quote 3"}
{modifier4="quote 4"} {modifier5="quote 5"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 2.16.

**Table 2.16** Modifiers Associated with the FILE Keyword in ANIMATE for HUMAN Data

Modifier	Description
CONNECT	Tecplot connectivity file (not needed with river line plot)
COORDS	Coordinate data (This defines location (ID,X,Y))
HEADER	Header file created by the HUMAN code

Modifier	Description
DETAIL	Detailed data
RESULTS	Output file used for Tecplot visualization

## 2.8.4 SOLUTION Keyword

The SOLUTION is used to define the solution type for which data will be processed. The following is this keyword's syntax:

```
SOLUTION [ "quote 1" ]
```

The single quote string is case sensitive. The modifiers associated with the SOLUTION keyword are given in Table 2.17.

**Table 2.17** Modifiers Associated with the SOLUTION Keyword in ANIMATE for HUMAN Data

Modifier	Description
ANADOSE	Analyte dose
ANARISK	Analyte risk
ANAHQ	Analyte hazard quotient
DOSEING	Ingestion dose
DOSEINH	Inhalation dose
DOSEEXT	External dose
DOSEDER	Dermal dose
HQING	Ingestion hazard quotient
HQINH	Inhalation hazard quotient
HQDER	Dermal hazard quotient
POPDOSE	Population dose (radioactive)
POPRISK	Population risk (radioactive)
RISKING	Ingestion risk
RISKINH	Inhalation risk
RISKEXT	External risk
RISKDER	Dermal risk
SUMDOSE	Dose summed over analyes
SUMRISK	Risk summed over analyes
SUMHQ	Hazard quotient summed over analyes

An example of this keyword is the following:

```
SOLUTION "ANARISK"
```



## 2.8.5 UNITS Keyword

The UNITS keyword is used to define the output units for the animation. The following is this keyword's syntax:

```
UNITS OUTPUT "quotel" FACTOR value1
```

The quote string is the desired output units, and the value is the units conversion factor to translate from the units output by the original code to the desired output units for the animation. An example of this keyword is the following:

```
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9 !Convert Ci/m^3 to pCi/L
```

## 2.8.6 Example Keyword File for HUMAN Data

An example keyword file for running an animation for HUMAN detailed data is shown in Table 2.18.

**Table 2.18** Example Keyword File for ANIMATE for HUMAN Data

```
REPORT "Test_Animate_HUMAN_UPLAND.rpt"
TITLE "Animation Test for HUMAN UPLAND locations"
USER "Carmen Arimescu"
OS WINDOWS !UNIX ! This matches information in the header file
VERBOSE
LOCATION UPLAND ! RIPARIAN or AQUATIC
!-----
CATEGORY HUMAN !---> Only one category of data allowed per run of the code
FILE HEADER "Human_CA_Ref_A_UpDrink_SACVIEW.Hdr" ! Header file
FILE DETAIL "Human_CA_Ref_A_UpDrink_Dtl.csv" ! Risk detailed data
! Files needed
FILE RESULTS "H3_UpDrink_out.dat" !Output file use for tecplot visualization
FILE CONNECT "All_connectivity.txt" ! TecPlot connectivity file
FILE COORDIN "All_locations.csv" ! Coordinates for the connectivity (This
defines location (ID,X,Y))
! Pick one or more times
TIMES ALL
! Pick a single analyte
ANALYTE "H3"
! Pick an analyte type
ANALTYPE "CONMEDIA"
! Pick a solution
SOLUTION "GWAT"
! Pick a single realization to animate
PICK_REL 1
! Set the output units
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9 !Convert Ci/m^3 to pCi/L
LEGEND TITLE = "H3 UpDrink" CONTOUR = "Concentration"
!
END
```



## 3.0 Back\_MASS2 – Columbia River Background Data Generation

### 3.1 Overview

A general equation for calculating background concentrations in surface water applies to all analytes irrespective of whether they have any significant contribution from atmospheric fallout from nuclear weapons testing. For simplicity, contributions from fallout are considered only in 1990 or later years. The following governing equation for the background concentration in surface water is used for all years before 1990:

$$C_R = C_B$$

The following equation is used for 1990 or later years:

$$C_R = C_B + M_B e^{(-\lambda_B [\text{Year} - 1990])}$$

where

- $C_R$  = Concentration of the analyte in surface water ( $\text{Ci}/\text{m}^3$  or  $\text{kg}/\text{m}^3$ )
- $C_B$  = Nominal background concentration ( $\text{Ci}/\text{m}^3$  or  $\text{kg}/\text{m}^3$ ) from natural sources
- $M_B$  = Concentration ( $\text{Ci}/\text{m}^3$  or  $\text{kg}/\text{m}^3$ ) from manmade sources (weapons testing fallout)
- $\lambda_B$  = Decay term (1/yr) that includes the effect of radioactive decay and leaching from the land surface into surface waters for the manmade sources

The values for  $C_B$ ,  $M_B$ , and  $\lambda_B$  have different values for every analyte. The variables  $C_B$  and  $M_B$  are defined as stochastic variables. The variable for  $\lambda_B$  is defined as a constant. The values for  $C_B$  and  $M_B$  can be set to a constant, including the constant zero. The value for  $\lambda_B$  can also be set to zero.

Representative values (units of  $\text{Ci}/\text{m}^3$ ) in the governing concentration equation for a few analytes are provided in Table 3.1

**Table 3.1** Representative Coefficient Values for Background Concentrations in the Columbia River

Analyte ID	$C_B$	$M_B$	$\lambda_B$	Fallout?
C14	$5.3 \times 10^{-11}$	0	0	No
Cl36	0	0	0	No
Cs137	0	$1.07 \times 10^{-11}$	0.023	Yes
H3	$3.9 \times 10^{-8}$	$3.04 \times 10^{-8}$	0.0562	Yes
I129	0	$1.28 \times 10^{-14}$	$4.41 \times 10^{-8}$	Very small
Tc99	0	$4.41 \times 10^{-11}$	$3.28 \times 10^{-6}$	Yes
U238	$1.9 \times 10^{-10}$	0	0	No

The stochastic distributions associated with the nonzero coefficients defined in Table 3.1 are defined using the following rules:

- **Variable  $C_B$ .** The triangular distribution will be used for all values of  $C_B$ . The distribution will be symmetric about the midpoint, and the half-range will be 50% of the mid-point. The variable tag is CB.

- **Variable  $M_B$ .** The triangular distribution will be used for all values of  $C_B$ . The distribution will be symmetric about the midpoint, and the half-range will be 50% of the mid-point. The variable tag is MB.
- **Variable  $\lambda_B$ .** This variable is always defined as a constant. For ease of input, a stochastic variable definition will be used. The variable tag is LB.

### 3.1.1 Location in the Processing Sequence

The Back\_MASS2 code is a preprocessor to the river code MASS2. The system controller (ESP) reads a file containing stochastic information (nominally named “biota.stoch”) and the ESD keyword file and then writes a series of “biota.key” files (one for each analyte by realization combination) that are processed one at a time by the Back\_MASS2 code. The Back\_MASS2 code writes a series of input files that provide background data (for the specific analyte and realization combination) for MASS2. Excerpts from a biota.stoch file are provided in Table 3.2. This file is not read by Back\_MASS2; however, it provides the basis for the files prepared for Back\_MASS2 by the ESP code.

**Table 3.2** Excerpts from a biota.stoch file

```
! Stochastic Keyword file for the MASS2 & Back_MASS2 codes
! Purpose:
!   This keyword file is for use in the ESP program. The ESP reads this
!   file and the ESD keyword file and writes a series of "biota.key" files
!   for use by the Back_MASS2 code. The Back_MASS2 code writes a series of
!   input files that provide stochastic background data for MASS2.
!   Background from fallout is modeled from 1990 forward
TITLE "Stochastic Keyword Data for MASS2 and Back_MASS2 for the CA Analysis"
SEED 4436546
DEBUG STOCHASTIC="stoch/biota_stoch.out"
!
! River IDs and names for use in the MASS2 background data
RIVER ID="PRD" NAME="Columbia River at Priest Rapids Dam"
! Suspended sediment loading (kg/m^3) in each river
STOCHASTIC BACKGROUND SOLUTION="sediment" RIVER="PRD" 1 0.00375
!-----| Median Value Distributions |
! Dissolved contaminant concentrations by analyte and river
! Units are Ci/m^3 for radionuclides and kg/m^3 for chemicals
STOCHASTIC BACKGROUND ANALYTE="C14" SOLUTION="CB" RIVER="PRD"
1 5.300E-11 "Natural background in Ci/m^3"
STOCHASTIC BACKGROUND ANALYTE="C14" SOLUTION="MB" RIVER="PRD"
1 0 "Multiplier on leachout term (unitless)"
STOCHASTIC BACKGROUND ANALYTE="C14" SOLUTION="LB" RIVER="PRD"
1 0 "Leachout removal constant (1/yr)"
!
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="CB" RIVER="PRD"
1 0 "Natural background in Ci/m^3"
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="MB" RIVER="PRD"
1 1.070E-11 "Multiplier on leachout term (unitless)"
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="LB" RIVER="PRD"
1 2.300E-02 "Leachout removal constant (1/yr)"
!
END
```

### 3.1.2 How the Code Is Invoked

Back\_MASS2 can run under either the Windows or the Linux operating systems. Under the Windows operating system (Releases 2000 or XP), Back\_MASS2 executes in a DOS box. A run of Back\_MASS2 is initiated by entering the following command line:

```
back_mass2 "Keyfilename"
```

Under the Linux operating system CSCALE is executed through any of the following Bourne Shell or C Shell commands:

```
back_mass2-1.exe "Keyfilename"
```

For these commands, “Back\_MASS2.EXE” or “back\_mass2-1.exe” is the name of the executable program and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If Back\_MASS2 is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If Back\_MASS2 cannot find or open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

### 3.1.3 Memory Requirements

The Back\_MASS2 code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. However, the code uses minimal memory because it operates only on a small amount of scalar data.

## 3.2 File Definitions

The Back\_MASS2 code reads one file. It writes a report file and three additional files for every river defined in the keyword file.

### 3.2.1 Input Files

The only input file for the Back\_MASS2 code is a control keyword file. An example keyword file Back\_MASS2 for carbon-14 is provided in Table 3.3. Descriptions for the keywords are provided in Section 3.3.

**Table 3.3** Example Keyword File for Back\_MASS2

\$ ESP TEMPLATE File	: MASS2/Biota.Stoch
\$ ESP Date Generated	: 06-28-2004
\$ ESP Program Name	: ESP-Unix
\$ ESP Version Number	: 1.2
\$ ESP Program Date	: 02 Feb 2004
\$ Current Run ID	: 20040628083102
TITLE "2004 Composite Analysis 10,000-year (Median Inputs) Assessment"	
USER "Eslinger-DWE-WEN"	
! Realization	
REALIZATION CURRENT=1 TOTAL=1	

```
! Analyte
ANALYTE ID="C14" TYPE="NR" NAME="Carbon-14" HALFLIFE=5715.000
! Time period for the analysis
PERIOD START=1944 STOP=12060 CLOSURE=2070
! River IDs and Names
RIVER ID="PRD" NAME="Columbia River at Priest Rapids Dam"
! Dissolved -- Stochastic (#Analytes x #Rivers)
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="CB" VALUE=5.300000E-11
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="MB" VALUE=0.000000E+00
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="LB" VALUE=0.000000E+00
! SEDIMENT -- Stochastic (#Rivers)
SEDIMENT RIVER="PRD" VALUE=3.750000E-03
END
```

### 3.2.2 Output Files

The Back\_MASS2 code writes a report file and three additional files for every river defined in the keyword file. The report file contains summary information for the code run and is not described further here. Examples of the three additional output files written for use in MASS2 are provided. The files contain concentrations in the water column (Table 3.4), concentrations in particulate matter (Table 3.5), and sediment loading (Table 3.6). The first two lines in each of these tables is actually one line in the data file, but the line is so long that it wraps to the next line in the word processor. Many of the entries in Table 3.4 were deleted to shorten the file for presentation.

**Table 3.4** Example Dissolved Concentration File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming dissolved concentration
for Cs137 at Columbia River at Priest Rapids Dam
01-01-1943 00:00:00 0.00000E+00 /
01-01-1989 00:00:00 0.00000E+00 /
01-01-1990 00:00:00 1.07000E-11 /
01-01-1991 00:00:00 1.04567E-11 /
01-01-1992 00:00:00 1.02189E-11 /
01-01-1993 00:00:00 9.98659E-12 /
...
01-01-5413 00:00:00 0.00000E+00 /
01-01-5414 00:00:00 0.00000E+00 /
...
01-01-12061 00:00:00 0.00000E+00 /
```

**Table 3.5** Example Particulate Concentration File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming particulate concentration for
Cs137 at Columbia River at Priest Rapids Dam
01-01-1943 00:00:00 0.0 /
01-01-12061 00:00:00 0.0 /
```

**Table 3.6** Example Sediment Loading File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming Sediment (kg/m^3) for
Columbia River at Priest Rapids Dam
01-01-1943 00:00:00 3.75000E-03 /
01-01-12061 00:00:00 3.75000E-03 /
```

### 3.3 Keyword Definitions for Back\_MASS2

#### 3.3.1 ANALYTE Keyword for Back\_MASS2

The ANALYTE keyword identifies an analyte for use in the MASS2 code. Only one ANALYTE keyword is allowed for a run of the Back\_MASS2 code. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"] [TYPE="quote 2"] [NAME="quote 3"] {HALFLIFE=N1}
```

The modifiers associated with the ANALYTE keyword are defined in Table 3.7.

**Table 3.7** Modifiers Associated with the ANALYTE Keyword in Back\_MASS2

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte identification string up to six characters in length. The analyte identification string is case sensitive, and spaces or hyphens change the definition. The analyte ID must match one of the analytes defined in the ESD keyword file.
TYPE	The quote string associated with the TYPE modifier string is a two-character analyte type indicator. The following are the valid entries for this string: <ul style="list-style-type: none"> <li>• NR – if the analyte is a radioactive element or an inorganic compound containing a radionuclide</li> <li>• NS – if the analyte is a stable (nonradioactive) element or inorganic compound</li> <li>• OR – if the analyte is an organic compound containing a radionuclide</li> <li>• OS – if the analyte is an organic compound, containing a stable (nonradioactive) elemental analyte or compound</li> </ul>
NAME	The quote string associated with the NAME modifier is an analyte name or description up to 72 characters in length.
HALFLIFE	The numerical entry associated with the HALFLIFE modifier is the half-life of the analyte. This value has units of years. Entry of this modifier is necessary when defining a radioactive analyte but should be omitted for nonradioactive analytes.

The following ANALYTE keyword defines the analyte carbon-14 (not organic, radioactive) with a half-life of 5715 years.

```
ANALYTE ID="C14" TYPE="NR" NAME="Carbon-14" HALFLIFE=5715.000
```

#### 3.3.2 DISSOLVED Keyword for Back\_MASS2

The DISSOLVED keyword identifies the background concentrations for an analyte in the water column (at the upstream boundary) in a river used in the MASS2 code. The dissolved concentrations for multiple rivers are defined by making multiple entries of the DISSOLVED keyword. The following is this keyword's syntax:

```
DISSOLVED [ANALYTE="Quote1"] [RIVER="Quote2"]  
[SOLUTION="Quote3"] [VALUE=N1]
```

The ID for the analyte is defined in a quote string associated with the modifier ANALYTE and is limited to six characters in length (see the ANALYTE keyword in Section 3.3.1). The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length (see the RIVER keyword in Section 3.3.6). The quote string associated with the modifier SOLUTION takes on one of three values: CB, MB, or LB. The definitions of these parameters for the background solution are defined in Section 3.1. The following three example DISSOLVED keywords define the background concentration for carbon-14 in the Columbia River at Priest Rapids Dam.

```
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="CB" VALUE=5.300000E-11  
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="MB" VALUE=0.000000E+00  
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="LB" VALUE=0.000000E+00
```

Concentrations may be defined for more than one river in a single run of the Back\_MASS2 code. A suite of three DISSOLVED keywords are required to define the background concentration for each river.

### 3.3.3 END Keyword for Back\_MASS2

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 3.3.4 PERIOD Keyword for Back\_MASS2

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
PERIOD [START=N1] [STOP=N2]
```

The modifier START and the associated value N1 identify the year for the start of the simulation period. The modifier STOP and the associated value N2 identify the year for the end of the simulation period. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example PERIOD keyword that simulates from 1944 through 3050:

```
PERIOD START=1944 STOP=3050
```

### 3.3.5 REALIZAT Keyword for Back\_MASS2

The REALIZAT (or REALIZATION) keyword identifies the current realization number and the total number of realizations to be simulated. The following is this keyword's syntax:

```
REALIZATION [CURRENT=N1] [TOTAL=N2]
```

The number of the current realization is identified by the modifier CURRENT. The total number of realizations for the analysis definition is identified by the modifier TOTAL. The following is an example REALIZAT keyword requesting Back\_MASS2 to process data for realization 23 out of 100 realizations:

```
REALIZAT CURRENT=23 TOTAL=100
```



### 3.3.6 RIVER Keyword for Back\_MASS2

The RIVER keyword identifies the ID and name of a river used in the MASS2 code. Multiple rivers are defined by making multiple entries of the RIVER keyword. The following is this keyword's syntax:

```
RIVER [ID="quote 1"] [NAME="quote 2"]
```

The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length. The descriptive name of the river is defined in the quote string associated with the modifier NAME and is limited to 48 characters in length. The following is an example RIVER keyword defining the Columbia River at Priest Rapids Dam.

```
RIVER ID="PRD" NAME="Columbia River at Priest Rapids Dam"
```

### 3.3.7 SEDIMENT Keyword for Back\_MASS2

The SEDIMENT keyword identifies the suspended sediment loading for water (at the upstream boundary) in a river used in the MASS2 code. The sediment loading for multiple rivers is defined by making multiple entries of the SEDIMENT keyword. The following is this keyword's syntax:

```
SEDIMENT [RIVER="quote"] [VALUE=N1]
```

The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length (see the RIVER keyword in Section 3.3.6). The suspended sediment loading (in  $\text{kg/m}^3$ ) in the river is defined in the numerical value associated with the modifier VALUE. The following is an example SEDIMENT keyword defining the suspended sediment loading in the Columbia River at Priest Rapids Dam.

```
SEDIMENT RIVER="PRD" VALUE=3.750000E-03
```

### 3.3.8 TITLE Keyword for Back\_MASS2

The TITLE keyword is used to define a single-line problem title. The problem title will be written to the output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the Back_MASS2 code."
```

### 3.3.9 USER Keyword for Back\_MASS2

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

## **4.0 CONSUME – Ecological Species Consumption Data**

### **4.1 Overview**

The CONSUME program reads a text file containing ecological species information, including a full predation matrix, and reformats the data into CONSUME keyword records for use in ECEM (see Eslinger et al. [2006b], Volume 2, Section 4.0). The predation matrix (the fraction of the intake for an ecological specie that comes from consuming other species) is also checked for errors during this process. Typically, the underlying data for ECEM are maintained in a spreadsheet. This program converts data from the spreadsheet into a format that can be inserted into an ECEM keyword file.

#### **4.1.1 Location in the Processing Sequence**

The CONSUME program is a preprocessor to the ECEM code.

#### **4.1.2 Memory Requirements**

The CONSUME code uses dynamic memory allocation. It is expected that most, if not all, of the runs of the CONSUME code will require less than 2 MB of memory.

### **4.2 File Definitions**

The CONSUME code reads one file and writes one file.

#### **4.2.1 Input Files**

The CONSUME code reads a text data file containing ecological species information in a comma-separated format. An example input data file for 12 ecological species is provided in Table 4.1. The first line in the file is an integer ( $N_S$ ) giving the number of species to process. This input is followed by a definition line for each of the  $N_S$  species. Each line contains the following:

- The species ID (up to 6 characters)
- The type of the species; valid entries for this string are
  - TA – if the species is a terrestrial animal
  - TP – if the species is a terrestrial plant
  - QA – if the species is an aquatic animal
  - QP – if the species is an aquatic plant.
- The species location; valid entries for this string are
  - RIPARIAN – riparian zone (river shore)
  - UPLAND – upland
  - AQUATIC – river.
- The species long name (up to 72 characters).

The definition lines are followed by consumption lines for each of the  $N_S$  species. The order of the consumption lines must match with the order in which the species are defined in the file. Each line contains  $N_S+2$  entries. The first  $N_S$  entries give the fraction of the intake for the species that comes from each of the other species. Valid entries are in the range 0 to 1. Plants always have all zeros, and animal species typically have entries that sum to 1. An exception to this rule is the adult salmon that moves up the river without consuming any food. The entry in the  $N_S+1$  position is the amount of sediment consumed by the species, expressed as a fraction of intake. This value should be zero for all species that are not aquatic animals. Valid values are in the range 0 to 1. The entry in the  $N_S+2$  position is the amount of soil consumed by the species, expressed as a fraction of intake. This value should be zero for all species that are not terrestrial animals. Valid values are in the range 0 to 1.

**Table 4.1** Example Input File for CONSUME

```

12
"RCTWOD", "TP", "RIPARIAN", "black cottonwood"
"RCRESS", "TP", "RIPARIAN", "Columbia yellowcress"
"RSEDGE", "TP", "RIPARIAN", "dense sedge"
"RFERN ", "TP", "RIPARIAN", "fern"
"RFUNGI", "TP", "RIPARIAN", "fungi"
"HRVMSE", "TA", "RIPARIAN", "Harvest mouse"
"RMLBRY", "TP", "RIPARIAN", "mulberry"
"RREEDC", "TP", "RIPARIAN", "reed canarygrass"
"RRUSHS", "TP", "RIPARIAN", "rushes"
"RSWCLV", "TP", "RIPARIAN", "Sweet Clover (Melalotus alba)"
"RARTPD", "TA", "RIPARIAN", "terrestrial arthropods"
"RRTULE", "TP", "RIPARIAN", "tule"
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0.3,0,0,0,0,0.3,0,0.1,0.3,0,0,0.02
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.1,0.1,0.1,0.1,0.1,0,0.2,0.1,0.1,0,0,0.1,0,0.5
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

```

## 4.2.2 Output Files

The CONSUME code writes one data file. This file is a text file that is always named “consume.rpt” and contains the reformatted consumption data, an echo of the predation matrix, and any error messages. Excerpts from a report file are provided in Table 4.2. The typical mode of operation is to extract the CONSUME keyword records from this file and paste them into an input file for the ECEM (see Eslinger et al. [2006b], Volume 2, Section 4.0) code.

**Table 4.2** Excerpts from a CONSUME Report File

```

!   Program: Consume
!   Version: 1.0.004
!   Revised: 30 Jan 2006
!   User:    Anonymous User
!   Date:    02-01-2006
!   Time:    12:12:33
!   File:    Consume_2006_02_01.Dat
!
CONSUME ID="RHMSE" PREY
"RSEDGE"  0.40000
"RREEDC"  0.40000
"RRUSHS"  0.10000
"RARTPD"  0.10000
SOILING   0.02000
!
CONSUME ID="RARTPD" PREY
"RCTWOD"  0.05000
"RCRESS"  0.05000
"RSEDGE"  0.05000
"RFUNGI"  0.05000
"RGRASS"  0.30000
"RMLBRY"  0.05000
"RONION"  0.02500
"RREEDC"  0.25000
"RRTVEG"  0.02500
"RRUSHS"  0.05000
"RRTULE"  0.05000
"RWILOW"  0.05000
SOILING   0.05400

```

### 4.3 Code Execution

CONSUME can run under either the Windows or Linux operating system. Under the Windows operating system (Releases 2000 or XP), CONSUME executes in a DOS box. A run of CONSUME is initiated by entering the following command line:

```
CONSUME "Data File"
```

Under the Linux operating system CONSUME is executed through any of the following Bourne Shell or C Shell commands:

```
consume-1.exe "Data File"
```

For these commands, “CONSUME.EXE” or “consume-1.exe” is the name of the executable program. One argument (file name) must be provided on the command line. The argument is the name of the data file to be processed.



## **5.0 CRGRAB – Columbia River Data Grabber**

### **5.1 Overview**

The CRGRAB (Columbia River Data Grabber) data extractor allows the user to collect and summarize the analyte mass or activity inputs to the Columbia River.

#### **5.1.1 Location in the Processing Sequence**

The CRGRAB program reads data files written by the GWDROP code. These are the same data files that the MASS2 code uses to input the analyte source term for transport in the river model. Data for more than one analyte or realization can be gathered in the same run of CRGRAB.

#### **5.1.2 How the Code Is Invoked**

CRGRAB runs only under the Linux operating system. CRGRAB is executed through any of the following Bourne Shell or C Shell commands:

```
crgrab-1.exe "Keyfilename"
```

For these commands, “crgrab-1.exe” is the name of the executable program, and “Keyfilename” is the name of the CRGRAB control keyword file. Both the name of the executable program and the keyword file may contain path information. If CRGRAB is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If CRGRAB cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **5.1.3 Memory Requirements**

The CRGRAB code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the CRGRAB code will require less than 256 MB of memory.

### **5.2 File Definitions**

The CRGRAB code reads two keyword files, a small number of MASS2 grid coordinate files, and potentially a large number of files containing inputs for MASS2. The code writes two or three output files, depending on the case definition.

#### **5.2.1 Input Files**

The CRGRAB code reads the ESD keyword file and a CRGRAB keyword file. In addition, CRGRAB reads a number of MASS2 grid coordinate files, and potentially a large number of files containing inputs for MASS2. The user modifies only the CRGRAB keyword file: the other files are read only for supporting information.

### 5.2.1.1 CRGRAB Keyword File

The CRGRAB keyword file has a name specified by the user and contains control information for extracting data. An example file is provided in Table 5.1. Detailed definitions of the keywords are provided in Section 5.3. The actual run of the code uses grid files with extensions “.000” through “.056”. Some of the file names were deleted from the example for presentation purposes.

**Table 5.1** Example Keyword File for the CRGRAB Code

```
TITLE "Test the CrGrab code for Reference Case A"
USER "Paul W. Eslinger"
FILE REPORT "CrGrab_New_Flow.rpt"
FILE ESD "c:\projects\sac\codes_1\crgrab\ESD_CA_Ref_A.key"
BASEPATH "x:\CA_Ref_A\"
GRABPATH "c:\projects\sac\codes_1\crgrab\"
ANALYTE LIST "H3" "Cl4" "Cl36" "Se79" FLOW
REALIZATION LIST 1
IGNORE 2000
TIME ANNUAL TOTAL
EXTRACT INFLUX
VERBOSE
LOCATION TOTAL !ANNUAL 1990 2000
! MASS2 Grid Coordinate Files
MASS2GRID
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.000"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.001"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.002"
  ...
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.054"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.055"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.056"
END
```

### 5.2.1.2 Other Input Files

CRGRAB reads the ESD keyword file to obtain analyte information and optionally analysis path information. The format of this file is described by Eslinger et al. (2006a, Section 2.0), so it is not discussed in any further detail here. CRGRAB also reads a number of MASS2 grid coordinate files. The names of these files are provided in the CRGRAB keyword file.

In addition, CRGRAB reads potentially a large number of files containing mass flux inputs for MASS2. For example, data for one realization of cesium-137 in the 2004 Composite Analysis was spread across approximately 1450 data files. The general format of these files is the following:

- Line 1: Header line
- Line 2: time, cumulative value
- Line 3: time, cumulative value
- ...
- Line n: time, cumulative value



The time format is mm-dd-yyyy hh:mm:ss, where the year can be four, five, or six digits long, if necessary. The cumulative value has units of kg if the file contains releases for a nonradioactive analyte, units of curies if the file contains releases for a radioactive analyte, and units of cubic meters if the file contains water volumes.

The naming convention for the river release files is built into CRGRAB. A typical file name associated with an analyte (cesium-137, for example, without path name) is "TMS-14219-Cs137.DAT". In general, the file name is built from the string "TMS-" followed by a CFEST node number ID, followed by a dash and the analyte ID, followed by the extension ".DAT". When data for water flux are extracted, the file name uses the character string "FLOW" instead of the analyte ID.

## 5.2.2 Output Files

The CRGRAB code writes two to three output files, depending on the case definition. The report file is named by the user and is written for every run of the code. It contains summary information from the code run and is not described in further detail.

The annual release file contains annual and cumulative release to the river by year for every requested analyte and is always named "Influx\_Annual.csv". Excerpts from this file for the three analytes technetium-99, tritium, and uranium-238 are provided in Table 5.2. The file is written in comma-separated format to facilitate import into a spreadsheet or graphics program. The first line of the file is shown as wrapped in the table but actually occupies only one line of the file. The cumulative release up to the year 2000 is provided in a separate file, always named "Influx\_Annual\_Locs.csv", as shown in Table 5.3. In addition, the total release to the river by releasing location can be output under control of the LOCATION keyword in the file named "Influx\_Total\_Locs.csv". Excerpts from an example output file of total releases by location are provided in Table 5.4. When data for water flux are extracted (see the ANALYTE keyword below) the file names have an additional "\_FLOW" in them. For example, "Influx\_Annual.csv" becomes "Influx\_Annual\_Flow.csv".

**Table 5.2** Annual River Release File fom CRGRAB

"Realization", "Year", "Tc99 Annual", "Tc99 Cumul.", "H3 Annual", "H3 Cumul.", "U238 Annual", "U238 Cumul."						
1,1944,	9.18482E-11,	9.18482E-11,	1.18844E-09,	1.18844E-09,	1.12149E-18,	1.12149E-18
1,1945,	3.88735E-03,	3.88735E-03,	1.83568E-01,	1.83568E-01,	1.37958E-05,	1.37958E-05
1,1946,	1.98659E-02,	2.37533E-02,	9.15834E-01,	1.09940E+00,	5.14122E-03,	5.15501E-03
1,1947,	2.77167E-02,	5.14700E-02,	1.21544E+00,	2.31484E+00,	4.36316E-02,	4.87866E-02
1,1948,	3.26185E-02,	8.40885E-02,	1.35563E+00,	3.67048E+00,	7.57750E-02,	1.24562E-01
1,1949,	3.05275E-02,	1.14616E-01,	1.20208E+00,	4.87255E+00,	8.16468E-02,	2.06208E-01
1,1950,	3.02840E-02,	1.44900E-01,	1.21135E+00,	6.08390E+00,	1.08975E-02,	2.17106E-01
1,1951,	3.25044E-02,	1.77404E-01,	1.53951E+00,	7.62341E+00,	1.08975E-02,	2.28003E-01
1,1952,	3.50948E-02,	2.12499E-01,	2.03253E+00,	9.65593E+00,	1.09275E-02,	2.38931E-01
1,1953,	3.62911E-02,	2.48790E-01,	6.50826E+00,	1.61642E+01,	1.08975E-02,	2.49828E-01
1,1954,	3.87063E-02,	2.87496E-01,	6.79331E+00,	2.29575E+01,	1.09373E-02,	2.60766E-01
1,1955,	4.08380E-02,	3.28334E-01,	7.33052E+00,	3.02880E+01,	2.53792E-02,	2.86145E-01
1,1956,	4.03984E-02,	3.68733E-01,	1.00113E+01,	4.02994E+01,	2.54488E-02,	3.11594E-01
1,1957,	3.71861E-02,	4.05919E-01,	1.15777E+01,	5.18770E+01,	2.53792E-02,	3.36973E-01
1,1958,	2.34013E-02,	4.29320E-01,	1.26491E+01,	6.45261E+01,	2.53794E-02,	3.62352E-01
1,1959,	1.33992E-02,	4.42719E-01,	1.44867E+01,	7.90128E+01,	2.53793E-02,	3.87732E-01
...						
1,1997,	6.01274E-01,	2.54193E+01,	1.76168E+02,	2.89322E+04,	3.25530E-01,	3.39689E+00
1,1998,	2.72972E-01,	2.56923E+01,	1.39264E+02,	2.90715E+04,	3.25528E-01,	3.72242E+00
1,1999,	5.13283E-01,	2.62056E+01,	1.66100E+02,	2.92376E+04,	3.25531E-01,	4.04795E+00
1,2000,	5.34088E-01,	2.67397E+01,	1.69791E+02,	2.94074E+04,	1.44196E-01,	4.19215E+00

**Table 5.3** Cumulative Release File from CRGRAB

"Realization", "Year", "Tc99", "H3", "U238"
1, 2000, 2.67397E+01, 2.94074E+04, 4.19215E+00

**Table 5.4** Total Release by Releasing Location from CRGRAB

"Realization", "Year", "Easting", "Northing", "Tc99", "H3", "U238"
1, 2000, 574615.4, 154506.0, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574576.0, 154543.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574536.7, 154581.7, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574497.3, 154619.5, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574457.9, 154657.4, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574418.6, 154695.2, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574379.3, 154733.1, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574339.9, 154770.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574710.1, 154562.7, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574686.3, 154605.6, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574659.7, 154645.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1, 2000, 574629.6, 154683.8, 0.00000E+00, 0.00000E+00, 0.00000E+00
...
1, 2000, 582534.5, 147463.5, 1.05315E-06, 1.08056E-04, 5.93517E-09
1, 2000, 582371.4, 147186.1, 1.19300E-06, 1.16438E-04, 6.43102E-09
1, 2000, 582447.3, 147193.6, 1.35335E-06, 1.32088E-04, 7.29537E-09
1, 2000, 582516.3, 147201.0, 1.50082E-06, 1.46482E-04, 8.09034E-09
1, 2000, 582500.8, 146901.5, 1.38131E-05, 1.47514E-03, 5.95506E-08
1, 2000, 582537.4, 146916.5, 1.23623E-05, 1.31756E-03, 5.36062E-08
1, 2000, 582573.6, 146931.1, 1.10899E-05, 1.17955E-03, 4.83707E-08
1, 2000, 582632.1, 146611.0, 1.63086E-05, 2.08297E-03, 4.54770E-08
1, 2000, 582662.5, 146641.1, 1.00104E-05, 1.23631E-03, 3.06861E-08
1, 2000, 582691.9, 146670.5, 5.53580E-06, 6.26356E-04, 1.98411E-08
1, 2000, 582766.8, 146433.2, 3.93529E-06, 4.57893E-04, 4.48185E-09
1, 2000, 582796.3, 146445.6, 1.90956E-08, 7.61659E-07, 2.48644E-11
1, 2000, 582824.9, 146456.9, 2.15329E-08, 8.58875E-07, 2.80380E-11

## 5.3 Keyword Definitions for CRGRAB

Most of the keywords for the CRGRAB code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 5.3.1 ANALYTE Keyword for CRGRAB

The ANALYTE keyword is used to define the analytes to be used in the simulation. The following is this keyword's syntax:

```
ANALYTE [ALL | LIST="quote 1" ... "quote n"] {FLOW}
```

The analytes requested must be a subset of the analytes for which environmental data were computed and stored by the inventory, release, and transport modules. If the modifier ALL is present, then data is extracted for every analyte defined in the ESD keyword file. If the LIST modifier is present, then it must

be followed by one or more quote strings that contain analyte ID's. If the modifier FLOW is present, then water flux information will be extracted rather than mass (or activity) flux. The following example ANALYTE keyword specifies that data will be extracted for three analytes: tritium, iodine-129, and carbon-14.

```
ANALYTE LIST "H3" "I129" "C14"
```

The following example ANALYTE keyword specifies that water flow data associated with the three analytes tritium, iodine-129, and carbon-14 will be extracted rather than the activity flux.

```
ANALYTE LIST "H3" "I129" "C14" FLOW
```

### 5.3.2 BASEPATH Keyword for CRGRAB

The optional BASEPATH keyword is used to define the directory where the base ESD keyword file resides. If this keyword is not entered, then the path from the ESD keyword file is used. The following is this keyword's syntax:

```
BASEPATH "quote1"
```

The single quote string contains the pathname of a directory where the base ESD keyword file for the problem resides. The path must terminate with a "/" character. The operating system keyword (OS) in the ESD keyword file is used to determine which character should be used. An example use of the BASEPATH keyword is the following:

```
BASEPATH "/home/ANALYSIS5/CA_Ref_A/"
```

### 5.3.3 DEBUG Keyword for CRGRAB

The DEBUG keyword is used to activate dumping of intermediate calculations to the report file. It should be used sparingly and with only one or two realizations; otherwise, the report file size can exceed the maximum allowable size of 2.1 gigabytes. The following is this keyword's syntax:

```
DEBUG [modifier 1] {modifier 2} ... {modifier 5}
```

Multiple DEBUG records can be entered with combinations of modifiers, or a single record can be entered containing all of the modifiers. The modifiers can be entered in any order. Table 5.5 describes the modifiers associated with the DEBUG keyword.

**Table 5.5** Modifiers Associated with the DEBUG Keyword in CRGRAB

Modifier	Description
GRID	Intermediate outputs on grid-related data extractions and calculations
INFLUX	Intermediate outputs on reading and processing influx data
TIMES	Intermediate outputs on time-registering data
TOTAL	Intermediate outputs on calculations for total releases
LOCATE	Intermediate outputs on calculations for radioactive analytes

The following entries are examples of the use of the DEBUG keyword:

```
DEBUG TIMES INFLUX GRID
DEBUG TOTAL
```

### 5.3.4 END Keyword for CRGRAB

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 5.3.5 EXTRACT Keyword for CRGRAB

The EXTRACT keyword is used to define the type of data to be extracted. The following is this keyword's syntax:

```
EXTRACT [ INFLUX ]
```

The single modifier INFLUX indicates that contaminant influx to the river is to be extracted. The data extractor has not been upgraded to extract water volumes, although water volumes are sent to the river. An example use of the EXTRACT keyword is the following:

```
EXTRACT INFLUX
```

### 5.3.6 GRABPATH Keyword for CRGRAB

The GRABPATH keyword is used to define the directory where output files will be written. The following is this keyword's syntax:

```
GRABPATH "quote1"
```

The single quote string contains the pathname of a directory where the output files containing extracted data will be written. The path must terminate with a "/" character. An example use of the GRABPATH keyword is the following:

```
GRABPATH "/home/ANALYSIS4/CA1_median/crgrab/"
```

### 5.3.7 IGNORE Keyword for CRGRAB

The IGNORE keyword directs the CRGRAB code to ignore any data after a specified year. The following is this keyword's syntax:

```
IGNORE [N1]
```

The single numerical value entered on this keyword is an integer calendar year. All data after this year will be ignored. An example use of the IGNORE keyword is the following:

```
IGNORE 4000
```

### 5.3.8 LOCATION Keyword for CRGRAB

The LOCATION keyword is used to calculate the total release to the river by releasing location. The following is this keyword's syntax:

```
LOCATION {TOTAL} {ANNUAL [ALL | Year1 Year2 ... Yearn]}
```

The modifier TOTAL activates output of the total contaminant influx to the river over the simulation period (as modified by the IGNORE keyword). This information is output for every location that the groundwater model delivers input to the river model and is written to the file "Influx\_Total\_Locs.csv" (or "Influx\_Total\_Locs\_Flow.csv" if water flow outputs are selected).

The modifier ANNUAL invokes output of annual data for one or more years for every location that the groundwater model delivers input to the river model. The ANNUAL modifier must be accompanied by either the ALL modifier or a list of one or more calendar years. The modifier ALL indicates annual values will be written out for all simulated years. A list of years activates annual outputs only for the years in the list. The annual data are written to the file "Influx\_Annual\_Locs.csv" (or "Influx\_Annual\_Locs\_Flow.csv" if water flow outputs are selected). An example use of the LOCATION keyword that selects total outputs by location as well as the outputs for the two years 2006 and 2007 is the following:

```
LOCATION TOTAL ANNUAL 2006 2007
```

### 5.3.9 MASS2GRID Keyword for CRGRAB

The MASS2GRID keyword is used to identify the suite of grid files used in the MASS2 code. The following is this keyword's syntax:

```
MASS2GRID ["quote 1"] {"quote 2" ... "quote n"}
```

The name of each grid file is identified in a quote string. Typically the grid file names will include the pathname. At least one grid file is required. An example use of the MASS2GRID keyword for three grid files is the following:

```
MASS2GRID  
"/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.000"  
"/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.001"  
"/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.002"
```

### 5.3.10 ONLY Keyword for CRGRAB

The ONLY keyword narrows the focus of the data extraction to a single TMS (flux release) file. The following is this keyword's syntax:

```
ONLY ["quote1"]
```

The name of the single desired TMS file is identified in a quote string. An example use of the ONLY keyword is the following:

```
ONLY "/home/ANALYSIS4/CA1_median/mass2/C14/1/cfest/TMS-6499-C14.DAT"
```

### 5.3.11 REALIZAT Keyword for CRGRAB

The REALIZAT (or REALIZATION) keyword identifies the realizations to be processed. The following is this keyword's syntax:

```
REALIZATION [ ALL | LIST N1 N2 N3 ... ]
```

The realizations requested must be a subset of the analytes for which environmental data were computed and stored by the inventory, release, and transport modules. If the modifier ALL is present, then data is extracted for every realization defined in the ESD keyword file. If the LIST modifier is present, then it must be followed by one or more numerical values that identify the realizations to process. Example uses of the REALIZAT keyword to extract data for all realizations and for just realizations 2, 3, and 91 are the following:

```
REALIZATION ALL  
REALIZATION LIST 3 2 91
```

### 5.3.12 TIME Keyword for CRGRAB

The TIME keyword identifies whether data are output accumulated to calendar year or summed for the entire simulation. The following is this keyword's syntax:

```
TIME {ANNUAL} {TOTAL}
```

If the modifier ANNUAL is present, then data are output accumulated by calendar year. If the modifier TOTAL is present, then data are output summed for the entire simulation. At least one of the ANNUAL or TOTAL modifiers must be used. The following example keyword identifies that data are to be output on both an annual and a simulation total basis:

```
TIME ANNUAL TOTAL
```

### 5.3.13 TITLE Keyword for CRGRAB

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the CRGRAB code."
```

### **5.3.14 USER Keyword for CRGRAB**

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### **5.3.15 VERBOSE Keyword for CRGRAB**

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```





## 6.0 CSCALE – Concentration Data Scaling

### 6.1 Overview

The ECDA program creates environmental concentration data accumulator (ECDA) files and initializes the contents. Other programs (GWDROP, CRDROP, AIRDROP, RIPSAC, and SOIL) then fill in the concentration values for different media. The CSCALE utility program can be used to calculate concentration values in an ECDA file as a linear function of data in other ECDA files. The linear combination of data can be a function of time. The same linear combination is used for all media in a given file. Two methods can be used:

- **Method #1.** The data in the output ECDA file is a multiple of the data in a single input file. For example, uranium-238 is run as a radionuclide through the inventory, release, and transport codes of the SAC. This utility could then be used to write a concentration data file for uranium as an element (chemical) for further use in the impacts modules.
- **Method #2.** The data in the output ECDA file is the sum of multiples of the data in two input files. For example, uranium-235 and uranium-238 are run as radionuclides through the inventory, release, and transport codes of the SAC. This utility could then be used to write a concentration data file for uranium as an element (chemical) for further use in the impacts modules. The output concentration is the sum of uranium-235 times its multiplier and uranium-238 times its multiplier.

All ECDA files used by this utility program must have an analyte defined in the ESD keyword file, and the file must be set up by the ECDA program.

#### 6.1.1 Location in the Processing Sequence

This code must be run after all transport calculations have been finished. The transport calculations may include running the RIPSAC and SOIL codes, depending on the problem being analyzed.

#### 6.1.2 How the Code Is Invoked

CSCALE can run under either the Windows or the Linux operating systems. Under the Windows operating system (Releases 2000 or XP), CSCALE executes in a DOS box. A run of CSCALE is initiated by entering the following command line:

```
CSCALE "Keyfilename"
```

Under the Linux operating system CSCALE is executed through any of the following Bourne Shell or C Shell commands:

```
cscale-1.exe "Keyfilename"
```

For these commands, “CSCALE.EXE” or “cscale-1.exe” is the name of the executable program, and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If CSCALE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If CSCALE cannot find or open

the keyword file, then the code will terminate execution after writing an error message to the standard output device.

### 6.1.3 Memory Requirements

The CSCALE code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. However, the code uses minimal memory because it reads and writes only a single record at a time from each of the input and output ECDA files.

## 6.2 File Definitions

There are three or more input files for every run of the code. There are two output files for every run of the code.

### 6.2.1 Input Files

The input files contain control information, the concentration data to be converted, and a concentration index file. More than one input concentration data file can be used.

#### 6.2.1.1 Keyword Control File

The CSCALE program is controlled by the entries in a keyword file. Table 6.1 contains an example keyword file for the CSCALE code. Definitions for the individual keywords are provided in Section 6.3.

**Table 6.1** Example Keyword File for the CSCALE Code

```
!
! Report File (Must be first Entry)
REPORT "Test.Rpt"
!
! Other identification information
TITLE " SAC Rev. 1 version of the CScale Code for multiple input files"
USER  "Carmen Arimescu"
!
EXECUTE
!DEBUG
!
! File names
FILE MAP      "\Sac\AAAA\ECDA_Map.CSV"
FILE OUTPUT   "\SAC\AAAA\CON_U.DAT"
!
ANALYTE ID="U" UNITS="kg"
!
FILE INPUT    "\SAC\AAAA\CON_U235.DAT"
! Multipliers for the data in the first concentration file
! Per the units given above, this also converts Ci to kg
SCALE
  1944 0.001
  3050 0.5
  12050 0.001
```

```
!  
FILE INPUT "\SAC\AAAA\CON_U238.DAT"  
! Multipliers for the data in the second concentration file  
! Per the units given above, this also converts Ci to kg  
SCALE  
    1944  0  
    2194  0.00526044  
    4444  0.0513772  
    7194  0.104849  
    11944 0.190206  
!  
! End of Inputs  
END
```

#### 6.2.1.2 Input ECDA Files

At least one input concentration data (ECDA) file must be identified that will serve as the source file for computing the scaled concentrations. This code assumes that the file was generated by the ECDA program, and that it contains concentration data entered by other codes of the SAC computational suite. Multiple input concentration data files may be used.

#### 6.2.1.3 ECDA Record Map File

This input file is the record map file associated with the concentration data files. This file is an ASCII file written by the ECDA program. The same map file applies to all input and output concentration data files for a single code run.

### 6.2.2 Output Files

There are two output files for every run of the code. These files contain a report of the run progress and the scaled concentration data.

#### 6.2.2.1 Report File

The report file is an ASCII file containing information about the run of the CSCALE program. It is written for every run of the code. All error messages (except for those indicating the report file could not be opened) are written to this file.

#### 6.2.2.2 Concentration Data File

The resulting scaled concentration data file written by the code is in the same format as the input concentration data files. The header lines in the file have been modified to indicate the new concentration units. The data have all been scaled by the multipliers defined by the SCALE keyword (see Section 6.3.7).

## 6.3 Keywords Definitions for CSCALE

Most of the keywords for the CSCALE code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.
- A SCALE keyword entry must be associated with a FILE keyword entry that defines an input file. The scaling factors are associated with the input file by their placement in the keyword file. For example, the second entry of a SCALE keyword defines the scaling factors for the second input file.

### 6.3.1 ANALYTE Keyword for CSCALE

The ANALYTE keyword is used to define the output analyte name and its associated concentration units. The following is this keyword's syntax:

```
ANALYTE [ ID="quote1" ] [ UNITS="quote2" ]
```

The analyte ID and output units are entered in quote strings, which must be enclosed in double quotation marks. Units up to 2 characters long are supported.

Exactly one ANALYTE keyword is required for every run of the code. The modifiers associated with the ANALYTE keyword are given in Table 6.2.

**Table 6.2** Modifiers Associated with the ANALYTE Keyword

Modifier	Description
ID	The quote string (up to six characters in length) associated with this modifier identifies the output analyte.
UNITS	The quote string associated with this modifier identifies the concentration data units for the output data file. The only allowable units are "Ci" or "kg". These units correspond to concentrations of Ci/m <sup>3</sup> or kg/m <sup>3</sup> for water media and Ci/kg or kg/kg for soil or sediment media.

The following example supports converting radionuclide concentration in curies to concentration in kilograms for uranium:

```
ANALYTE ID="U" UNITS="kg"
```

### 6.3.2 DEBUG Keyword for CSCALE

The DEBUG keyword is used to initiate intermediate data output statements. The following is this keyword's syntax:

```
DEBUG
```

This keyword should not be used when a large data set is being processed because it may result in a very large output file. There are no modifiers or quote strings associated with the DEBUG keyword.

### 6.3.3 END Keyword for CSCALE

The END keyword signifies the end of all keyword data. It should be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

END

### 6.3.4 EXECUTE Keyword for CSCALE

The EXECUTE keyword signifies that the user wishes to perform problem execution. If this keyword is not entered, the inputs are checked for consistency, but the problem will not be executed. This is useful if the run being set up is expected to take a significant amount of computation time. The syntax for this keyword record is the following:

EXECUTE

### 6.3.5 FILE Keyword for CSCALE

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

FILE [modifier1= "quote 1"] {modifier2="quote 2"} {modifier3="quote 3"}

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 6.3.

**Table 6.3** Modifiers Associated with the FILE Keyword

Modifier	Description
INPUT	The quote string associated with this modifier identifies an input concentration data file. The input file is assumed to be a concentration file created by the ECDA code. This file contains a set of concentration data that are to be scaled. Every input file must be declared using a separate entry of the FILE keyword. Every entry of an input file must be followed by an associated SCALE keyword before the next input file is defined.
OUTPUT	The quote string associated with this modifier identifies the output concentration data file. This output file will be a file in the same format as the input concentration file or files; however, the concentration data will all be scaled by the associated multipliers, and the data unit definitions in the header lines will be modified.
MAP	The quote string associated with this modifier identifies the record number map associated with the input concentration data file. This file is an input file nominally generated by the ECDA code.

The following example entries form a complete set of FILE keywords for a run of the code where uranium-238 concentrations are converted to uranium concentrations:

```
FILE  OUTPUT "Human_U.dat"  
FILE  MAP  "Human_Map.csv"  
FILE  INPUT "Human_U238.dat"
```

The following example entries form a complete set of FILE keywords for a run of the code where uranium-235 and uranium-238 concentrations are converted to uranium concentrations and summed together:

```
FILE  OUTPUT "Human_U.dat"  
FILE  MAP  "Human_Map.csv"  
FILE  INPUT "Human_U235.dat"  
FILE  INPUT "Human_U238.dat"
```

### 6.3.6 REPORT Keyword for CSCALE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. An example REPORT keyword record is the following:

```
REPORT "/SAC/SystemCodes/ECEM/Test1.rpt"
```

### 6.3.7 SCALE Keyword for CSCALE

The SCALE keyword defines the scaling factors for the concentration data. The following is this keyword's syntax:

```
SCALE [value1, value2 ... valueN]
```

A SCALE keyword entry must be associated with a FILE keyword entry that defines an input file. The scaling factors are associated with the input file by their placement in the keyword file. For example, the second entry of a SCALE keyword defines the scaling factors for the second input file.

The following keyword record defines time-dependent scaling factors to use with an input data file.

```
SCALE !NUMBER=41 TABLE  
1944  0  
2194  0.00526044  
2694  0.0156986  
3694  0.0362475  
4694  0.0563674  
5694  0.0760673  
6944  0.100115  
8194  0.123536  
9444  0.146348  
10444 0.164169  
11944 0.190206
```

### **6.3.8 TITLE Keyword for CSCALE**

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, the program will error terminate. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the CScale code."
```

### **6.3.9 USER Keyword for CSCALE**

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, the program will error terminate. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```





## **7.0 ECDA – Environmental Concentration Data Accumulator Setup**

### **7.1 Overview**

The ECDA code creates the binary environmental concentration data accumulator (ECDA) files and initializes the contents for each environmental medium. In addition, the ECDA code creates the stochastic data libraries of soil distribution coefficients ( $K_d$ ) and river bank seep water dilution factors.

#### **7.1.1 Location in the Processing Sequence**

The ECDA code typically is run before any other code in an assessment. It must be run before any of the following codes can execute: AIRDROP, GWDROP, CRDROP, SOIL, RIPSAC, or any of the impacts codes.

#### **7.1.2 How the Code Is Invoked**

ECDA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), ECDA executes in a DOS box. A run of ECDA is initiated by entering the following command line:

```
ECDA "Keyfilename"
```

Under the Linux operating system, ECDA is executed through any of the following Bourne Shell or C Shell commands:

```
ecda-1.exe "Keyfilename"
```

For these commands, “ECDA.EXE” or “ecda-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and keyword file may contain path information. If ECDA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If ECDA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **7.1.3 Memory Requirements**

The ECDA program has minimal memory requirements. A run of the code to create files using 1000 locations, 100 realizations, and 15 analytes required less than 4 MB of memory.

### **7.2 File Definitions**

The ECDA program reads one input file and writes multiple output files. These files are described in the following sections.

## 7.2.1 Input Files for ECDA

The ECDA program reads the ESD keyword file that contains the basic control information for the entire assessment. An example file is provided in Table 7.1.

**Table 7.1** Example Keyword File for ECDA

```
! Example keywords for ECDA
REPORT "ESD_CPO.rpt"
TITLE "Central Plateau Optimization"
USER "PWE - WEN"
DEBUG ECDA
REALIZAT 1
PERIOD START=1944 STOP=3050 CLOSURE=2070
ANALYTE ID="I129" NAME="Iodine-129" TYPE="NR" AIR="IODINE" COMPUTE
  MOLWGT=1.289050E+02 HALFLIFE=1.570000E+07 SPECIFIC=1.767954E-04
  DFIMM=1.039647E-01 DFSIED=6.930000E-20 GAMMA=2.500000E-02
  MOLDIFF=1.050000E-05 GASDIFF=8.340000E-02 HENRY=1.033515E+02
TIMES 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990
!
ECHO KDSOIL ! Echo SORP-water Kd values to the ECDA report file
FILE KDSOIL NAME="/home/CPO/ecda/KDSOIL.dat" SEED=232323.0 !CREATE
KDSOIL ID="KDI" 1 0.0001316 "Soil-water Kd iodine (L/g)" UNITS="L/g"
KDSOIL ID="KDTc" 1 0.0 "Soil-water Kd technetium (L/g)" UNITS="L/g"
!
ECHO DILUTE ! Echo dilution data to the ECDA report file
FILE DILUTE NAME="/home/CPO/ecda/DILUTE_CPO.Dat" SEED=123445.0 CREATE
DILUTE ID="DF5m" 1 4.8303E-01 "Dilution factor" UNITS="none"
!
FILLECDA GWAT FIXED = 0.0
FILLECDA PWAT FIXED = 0.0
FILLECDA SWAT FIXED = 0.0
FILLECDA SEEP FIXED = -1.0
FILLECDA SEDI FIXED = 0.0
FILLECDA SORP FIXED = -1.0
FILLECDA SODR FIXED = -1.0
FILLECDA SOGW FIXED = -1.0
FILLECDA SOSW FIXED = -1.0
FILLECDA AIRC FIXED = 0.0
FILLECDA AIRD FIXED = 0.0
FILE HEADER NAME="/home/CPO/ecda/Sacview_CPO.hdr" CREATE
FILE I_ECDA NAME="/home/CPO/ecda/ECDA_CPO.map" CREATE
FILE C_ECDA ANALYTE="Tc99" NAME="/home/CPO/ecda/Tc99.dat" CREATE
FILE C_ECDA ANALYTE="I129" NAME="/home/CPO/ecda/I129.dat" CREATE
FILE C_ECDA ANALYTE="U" NAME="/home/CPO/ecda/U.dat" CREATE
!
LOCATION ID="UH0001" NAME="UnsuitableForAgricul"
  EASTING=576022 NORTHING=154367
  GWAT TYPE = "UPLAND" LOCUS = "HANFORD"
  APSD = 4.00E-02 POROSITY= 3.50E-01 FOC = 1.0E+00
  VEGCOV = 5.00E-01 NECF = 1.00E+00 RHOS = 1.50E+00
```

```

TEMP      = 2.85E+02  MSWIND  = 3.44E+00  MZWIND = 3.44E+00
IRG_SWAT  ="QHP025"   AREA    = 1457376.1  SRH    = 1.8E-02
!
LOCATION ID="RHP001" NAME="RiparianVernitaBridge"
EASTING=557379.2 NORTHING=144876.4
GWAT  TYPE="RIPARIAN" LOCUS="HANFORD" MILE=389.00
APSD   = 4.00E-02  COXYGEN = 1.10E-02  POROSITY= 3.50E-01
NECF   = 1.00E+00  RHOS    = 1.50E+00  SRH     = 1.8E-02
MZWIND = 3.44E+00  AREA    = 9360      VEGCOV  = 5.00E-01
MSWIND = 3.44E+00  FOC     = 1.0E+00    TEMP    = 2.85E+02
!
LOCATION ID="QHP025" NAME="AquaticBenton100BC"
EASTING=566217.6 NORTHING=145715.6 SWAT SEDI PWAT
TYPE="AQUATIC" LOCUS="HANFORD" MILE=383.20
COXYGEN = 1.10E-02 AREA = 2.5E+03
!
LOCATION ID="AH0001" NAME="Air 100 D Area" EASTING=574436.00
NORTHING=151194.98 GWAT SOGW SOSW SODR AIRC AIRD TEMP=2.85E+02
TYPE="UPLAND" LOCUS="HANFORD", IRG_SWAT ="QHP136" VEGCOV=5.00E-01
APSD   = 4.00E-02  POROSITY= 3.50E-01  FOC    = 1.0E+00
NECF   = 1.00E+00  RHOS    = 1.50E+00  SRH    = 1.8E-02
MZWIND = 3.44E+00  MZWIND  = 3.44E+00  AREA   = 2500
!
END

```

## 7.2.2 Output Files for ECDA

The ECDA code writes a number of output files. A report file is always written. The number of other files written depend on the input options. The following list summarizes the output files:

- **Report file.** A report file, named “ecda.rpt” is written for every run of the code. This file is a text file containing information about the run progress. Error messages are written to this file as well.
- **ECDA files.** A separate ECDA file is written for every analyte if the CREATE and C\_ECDA modifiers are present on the FILE keyword. The file format is described by Eslinger et al. (2006a, Section 2.2.1). These binary files contain the media concentrations calculated by the transport codes that are saved for use in the impacts codes.
- **Map index file.** This file is written if a file name is entered in the keyword file (FILE keyword, I\_ECDA modifier). The file format is described by Eslinger et al. (2006a, Section 2.2.2). This file contains indexing information to quickly locate data for specific times and locations in the binary ECDA files.
- **Header file.** This file is written if a file name is entered in the keyword file (FILE keyword, I\_ECDA modifier). The file format is described by Eslinger et al. (2006a, Section 2.2.3). This file is used to facilitate retrieving and displaying data.
- **Dilution file.** This file is written if the CREATE modifier is present in the keyword file (FILE keyword, DILUTE modifier). The file format is described by Eslinger et al. (2006a, Section 2.3.1). This file contains a library of generated dilution factors that describe the mixing of groundwater and river water in the riparian zone modeling that are used in the RIPSAC code.

- **Kdsoil file.** This file is written if the CREATE modifier is present in the keyword file (FILE keyword, KDSOIL modifier). The file format is described by Eslinger et al. (2006a, Section 2.3.2). This file contains a library of generated solid-aqueous partition coefficients ( $K_d$  values) that are used in a number of other SAC programs.

## 7.3 Keyword Definitions for ECDA

The ECDA code gets all of its control information from keywords in the ESD keyword file. In general, keywords for the ECDA code can be entered in any order. The only restriction is that the END keyword must be the last keyword in the file. The following keyword descriptions have been modified from those described by Eslinger et al. (2006a, Section 2.1) to restrict the definition to information used by the ECDA code.

### 7.3.1 ANALYTE Keyword for ECDA

The ANALYTE keyword is used to define the analytes to be used in the simulation. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"] [TYPE="quote 2"] [NAME="quote 3"]
```

A separate ANALYTE keyword must be entered for every analyte to be included in the simulation. Table 7.2 provides the modifiers associated with the ANALYTE keyword.

**Table 7.2** Modifiers Associated with the ANALYTE Keyword for ECDA

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte identification string up to six characters in length. The analyte identification string is case sensitive, and spaces or hyphens change the definition. All data in the analyte identification strings must satisfy the following conventions: <ul style="list-style-type: none"> <li>• Only the first entry in the analyte identification string is capitalized.</li> <li>• No embedded spaces or hyphens are used, even for radionuclides.</li> <li>• Individual elements are defined using the standard element abbreviation.</li> </ul>
TYPE	The quote string associated with the TYPE modifier string is a two-character analyte type indicator. The following are the valid entries for this string: <ul style="list-style-type: none"> <li>• NR – if the analyte is a radioactive element or an inorganic compound containing a radionuclide</li> <li>• NS – if the analyte is a stable (nonradioactive) element or inorganic compound</li> <li>• OR – if the analyte is an organic compound containing a radionuclide</li> <li>• OS – if the analyte is an organic compound, containing a stable (nonradioactive) elemental analyte or compound</li> </ul>
NAME	The quote string associated with the NAME modifier is an analyte name or description up to 72 characters in length.

The following ANALYTE keywords select the analytes tritium, uranium-235, and chemical uranium for analysis:

```
ANALYTE ID="H3"   NAME="Tritium"   TYPE="NR"  
ANALYTE ID="U235" NAME="Uranium-235" TYPE="NR"  
ANALYTE ID="U"   NAME="Uranium"   TYPE="NS"
```

### 7.3.2 DEBUG Keyword for ECDA

The DEBUG keyword initiates output of detailed information on the processing by the code. The following is this keyword's syntax:

```
DEBUG {ECDA}
```

This keyword has no effect if the modifier ECDA is not present. The following keyword will initiate debug outputs:

```
DEBUG ECDA
```

### 7.3.3 DILUTE Keyword for ECDA

The DILUTE keyword is used to enter the definition of a statistical distribution for stochastic water dilution variables used in the riparian zone water mixing model. The following is this keyword's syntax:

```
DILUTE [ID="quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2}  
      {LABEL="quote2"} [UNITS="quote3"]
```

The quote string associated with the ID modifier is a unique character string of up to 20 characters that will be used to identify this stochastic variable for subsequent uses. It is case sensitive, and embedded spaces are significant. The quote string associated with the optional modifier LABEL contains a description for the stochastic variable that can be up to 64 characters long. An entry for quote2 is not required, although it is used for labeling purposes if present. However, if the modifier LABEL is present, the associated quote string must be entered as well. The quote string associated with the UNITS modifier contains a units descriptor for the data. The strings "none" or "unitless" should be used if the variable is unitless. Section 45.0 contains further information about generating stochastic values.

A dilution factor that is triangular on the triple (0.2, 0.5, 0.99) could use the following keyword entry:

```
DILUTE ID="ID#1" 6 0.2 0.5 0.99 LABEL="Example dilution factor"  
      UNITS="none"
```

The stochastic values generated from information on the DILUTE keyword are used only in the RIPSAC code.

### 7.3.4 ECHO Keyword for ECDA

The ECHO keyword is used to initiate output of summary information code when processing the statistical distributions defined by KDSOIL and DILUTE keywords. The following is this keyword's syntax:

```
ECHO {KDSOIL} {DILUTE}
```

If the KDSOIL modifier is present then variable definitions and summary statistics on generated values will be written to the report file for every distribution specified on a KDSOIL keyword. If the DILUTE modifier is present, then variable definitions and summary statistics on generated values will be written to the report file for every distribution specified on a DILUTE keyword. Example uses of this keyword are the following:

```
ECHO KDSOIL
ECHO DILUTE
ECHO DILUTE KDSOIL
```

### 7.3.5 END Keyword for ECDA

The END keyword signifies the end of all keyword data. It should be the last keyword in the keyword file. Any data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

There are no modifiers or quote strings associated with the END keyword.

### 7.3.6 FILE Keyword for ECDA

The FILE keyword is used to enter the names of many of the files used in a simulation run. The following is this keyword's syntax:

```
FILE [NAME="quote 1"] {ANALYTE="quote 2"}
[ HEADER | KDSOIL | DILUTE | C_ECDA | I_ECDA ] {CREATE}
```

The file names must be entered with the complete path. One FILE keyword is required for every analyte for which concentrations are to be generated. Every file definition requires the entry of a separate FILE keyword. Table 7.3 provides the file type modifiers associated with the FILE keyword.

**Table 7.3** Modifiers Associated with the FILE Keyword for ECDA

Modifier	Description
NAME	The quote string associated with the NAME modifier is a file name (including path) up to 200 characters in length.
ANALYTE	If the file name is associated with the C_ECDA modifier, the ANALYTE modifier must also be entered to indicate which analyte is to be associated with the ECDA concentration data file.
HEADER	This modifier indicates that the FILE keyword is defining a header file that facilitates extraction of human-readable concentration data by post-processor programs.
KDSOIL	This modifier indicates that the FILE keyword is defining a file to contain stochastic realizations of all of the random variables defined using the KDSOIL keyword.
DILUTE	This modifier indicates that the FILE keyword is defining a file to contain stochastic realizations of all of the random variables defined using the DILUTE keyword.

Modifier	Description
C_ECDA	This modifier indicates that the FILE keyword is defining an ECDA concentration data file. Concentrations for each analyte are contained in separate files. The ANALYTE modifier is used to associate an analyte with this file.
I_ECDA	This modifier indicates that the FILE keyword is defining a record index file for mapping into all ECDA concentration files.
CREATE	When this modifier is present, the file will be created. If the CREATE modifier is not present, no actions occur for that file.

The following example entries define the concentration files for a suite of analytes:

```
FILE C_ECDA ANALYTE="H3" NAME="/test/ecda/H3_median.dat" CREATE
FILE C_ECDA ANALYTE="C14" NAME="/test/ecda/C14_median.dat" CREATE
FILE C_ECDA ANALYTE="I129" NAME="/test/ecda/I129_median.dat" CREATE
FILE C_ECDA ANALYTE="Tc99" NAME="/test/ecda/Tc99_median.dat" CREATE
```

If present, the CREATE flag causes the following actions:

- Deletion of any existing file by that name and creation of a new file.
- If the file is associated with the HEADER, DILUTE, KDSOIL, INFILT, or I\_ECDA modifiers, new data are written to the file.
- If the file is associated with the C\_ECDA modifier, the concentration data are initialized to the values identified by the FILLECDA keyword.

The following example entries define the files for soil-water  $K_d$  values and the water dilution factors for the river-shore module:

```
FILE KDSOIL NAME="KDSOIL.CSV" SEED=23232.0 CREATE
FILE DILUTE NAME="DILUTE.CSV" SEED=12345.0 CREATE
```

The following example entries define the Header file and ECDA record number index files:

```
FILE HEADER NAME="/test/ecda/Sacview_median.hdr" CREATE
FILE I_ECDA NAME="/test/ecda/ECDA_median.map" CREATE
```

### 7.3.7 FILLECDA Keyword for ECDA

The FILLECDA keyword controls filling ECDA files with fixed or random concentrations as they are initialized. The following is this keyword's syntax:

```
FILLECDA [GWAT|SWAT|AIRC|AIRD|PWAT|SEDI|SORP|SEEP|SODR|SOGW|SOSW]
[ FIXED=N1 | RANDOM ]
```

The value  $N_1$  is the single value to be used to initialize the ECDA concentration file for the specified media. Concentrations can be set to zero, or an invalid value (negative number). A separate FILLECDA keyword must be used for each media. Table 7.4 describes the modifiers associated with the FILLECDA keyword.

**Table 7.4** Modifiers Associated with the FILLECDA Keyword in the ESD File

Modifier	Description
GWAT	Presence of this optional modifier indicates that groundwater concentrations will be initialized with the specified value.
SWAT	Presence of this optional modifier indicates that surface water concentrations will be initialized with the specified value.
AIRC	Presence of this optional modifier indicates that atmospheric concentrations will be initialized with the specified value.
AIRD	Presence of this optional modifier indicates that atmospheric deposition rates will be initialized with the specified value.
PWAT	Presence of this optional modifier indicates that river bottom pore water concentrations will be initialized with the specified value.
SEDI	Presence of this optional modifier indicates that sediment concentrations (on the river bottom) will be initialized with the specified value.
SORP	Presence of this optional modifier indicates that riparian zone soil concentrations (on the land surface) will be initialized with the specified value.
SEEP	Presence of this optional modifier indicates that seep water concentrations (on the land surface) will be initialized with the specified value.
SODR	Presence of this optional modifier indicates that upland soil concentrations from non-irrigated soils will be initialized with the specified value.
SOGW	Presence of this optional modifier indicates that upland soil concentrations using groundwater for irrigation will be initialized with the specified value.
SOSW	Presence of this optional modifier indicates that upland soil concentrations using surface water for irrigation will be initialized with the specified value.
RANDOM	This modifier initializes the values to the media to a random number on the interval (0,1).
FIXED	The numerical value associated with this modifier is used to initialize all entries for the selected media.

The following example entries assign values to use in initialization of the ECDA files:

```

FILLECDA GWAT FIXED = 0.0
FILLECDA PWAT FIXED = 0.0
FILLECDA SWAT FIXED = 0.0
FILLECDA SEEP FIXED = -1.0
FILLECDA SEDI FIXED = 0.0
FILLECDA SORP FIXED = -1.0
FILLECDA SODR FIXED = -1.0
FILLECDA SOGW FIXED = -1.0
FILLECDA SOSW FIXED = -1.0
FILLECDA AIRC FIXED = 0.0
FILLECDA AIRD FIXED = 0.0

```

### 7.3.8 KDSOIL Keyword in ECDA

The KDSOIL keyword is used to enter the definition of a statistical distribution for the solid-aqueous distribution coefficient ( $K_d$ ) to be used for calculating soil concentrations from groundwater concentrations. The following is this keyword's syntax:



```
KDSOIL [ID="quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2}
{LABEL="quote2"} [UNITS="quote3"]
```

The quote string associated with the ID modifier is a unique character string of up to 20 characters that will be used to identify this stochastic variable for subsequent uses. It is case sensitive, and embedded spaces are significant. The quote string associated with the optional modifier LABEL contains a description for the stochastic variable that can be up to 64 characters long. An entry for quote2 is not required, although it is used for labeling purposes if present. However, if the modifier LABEL is present, the associated quote string must be entered as well. The quote string associated with the UNITS modifier contains a units descriptor for the data. The strings “none” or “unitless” should be used if the variable is unitless. Section 1.0 contains further information about generating stochastic values.

A  $K_d$  that is triangular on the triple (0.2, 0.5, 0.99) could use the following keyword entry:

```
KDSOIL ID="ID#1" 6 0.2 0.5 0.99 LABEL="Example Kd" UNITS= "L/g"
```

The data entered by this keyword are used in the river model MASS2 and in the riparian zone model RIPSAC.

### 7.3.9 LOCATION Keyword for ECDA

The LOCATION keyword identifies the locations where concentrations will be generated for use in the impacts modules. The following is this keyword’s syntax:

```
LOCATION [ID="quote1"] [EASTING=N1] [NORTHING=N2] {GWAT} {SWAT}
{AIRC} {AIRD} {PWAT} {SEDI} {SORP} {SEEP} {SODR} {SOGW} {SOSW}
[NAME="quote2"] {LOCUS="quote3"}]
```

Table 7.5 provides the modifiers and associated data for the LOCATION keyword.

**Table 7.5** Modifiers Associated with the LOCATION Keyword in the ESD File

Modifier	Description
ID	The location identification string is entered using the ID modifier. This string is limited to 6 characters and must be unique. It is used to associate other data with a specific location.
EASTING	This entry is associated with the easting coordinate for the location. These coordinates are expressed in terms of the Lambert projection of the Washington State Plane North American Datum of 1983, expressed in meters.
NORTHING	This entry is associated with the northing coordinate for the location. These coordinates are expressed in terms of the Lambert projection of the Washington State Plane North American Datum of 1983, expressed in meters.
NAME	The quote string associated with the NAME modifier contains a descriptive name of up to 64 characters in length that is used for labeling purposes.
LOCUS	The quote string associated with the LOCUS modifier is used in the MOBILEHOME code to identify a river bank boundary for ecological species. The two valid options are “HANFORD” and “FAR SIDE”.
GWAT	Presence of this optional modifier indicates that groundwater concentrations will be computed at this location.

Modifier	Description
SWAT	Presence of this optional modifier indicates that surface water concentrations will be computed at this location.
AIRC	Presence of this optional modifier indicates that atmospheric concentrations will be computed at this location.
AIRD	Presence of this optional modifier indicates that atmospheric deposition rates will be computed at this location.
PWAT	Presence of this optional modifier indicates that river bottom pore water concentrations will be computed at this location.
SEEP	Presence of this optional modifier indicates that seep water concentrations (on the land surface) will be computed at this location.
SODR	Presence of this optional modifier indicates that upland soil concentrations from non-irrigated soils will be initialized with the specified value.
SORP	Presence of this optional modifier indicates that riparian zone soil concentrations (on the land surface) will be computed at this location.
SEDI	Presence of this optional modifier indicates that sediment concentrations (on the river bottom) will be computed at this location.
SOGW	Presence of this optional modifier indicates that soil concentrations using groundwater for irrigation will be computed at this location.
SOSW	Presence of this optional modifier indicates that soil concentrations using surface water for irrigation will be computed at this location. Refer to the IRG_SWAT modifier to specify the surface water location.

The following example keywords define three impact locations:

```
LOCATION ID="HL0151" NAME="Upland location"
    EASTING=594737.5  NORTHING=127827.4  GWAT
LOCATION ID="HL0417" NAME="Riparian zone location"
    EASTING=557375.3  NORTHING=144885.2  GWAT SORP SEEP
LOCATION ID="HL0413" NAME="Richland municipal water intake"
    EASTING=595445.1  NORTHING=109753.5  SWAT PWAT SEDI
```

### 7.3.10 PERIOD Keyword for ECDA

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
PERIOD [START=year1] [STOP=year2] [CLOSURE=year3]
```

The modifier START and the associated value year<sub>1</sub> identify the start year for the simulation. The start of the simulation period must be 1944 or later, or the inventory code will error terminate. The modifier STOP and the associated value year<sub>2</sub> identify the end year for the simulation. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The modifier CLOSURE and the associated value year<sub>3</sub> identify the year that site closure occurs. The year of site closure cannot be smaller than the start year. The following example PERIOD keyword simulates from 1944 through 3050 with site closure occurring at 2050:

```
PERIOD START=1944 STOP=3050 CLOSURE 2050
```

### 7.3.11 REALIZAT Keyword for ECDA

The REALIZAT (or REALIZATION) keyword identifies the number of realizations to be simulated. The following is this keyword's syntax:

```
REALIZAT N1
```

The number of realizations is given by the single numerical entry N1. The valid number of realizations is 1 to 9999. Run times and disk storage requirements are directly proportional to the number of realizations. The following is an example REALIZAT keyword requesting 25 realizations:

```
REALIZAT 25
```

### 7.3.12 TITLE Keyword for ECDA

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the ECDA code."
```

### 7.3.13 USER Keyword for ECDA

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```



## **8.0 ECDA\_ASCII – Environmental Concentration Data Accumulator File Conversions**

### **8.1 Overview**

The ECDA\_ASCII program converts binary environmental concentration files into text format. The sole purpose of the conversion is to create a file that is human readable in a text editor.

#### **8.1.1 Location in the Processing Sequence**

The ECDA\_ASCII program can be executed anytime after the ECDA files have been created. However, other programs such as AIRDROP, GWDROP, CRDROP, RIPSAC, and SOIL write concentration data to these files. Thus, ECDA\_ASCII is usually run after all of the other codes have been executed.

#### **8.1.2 How the Code Is Invoked**

ECDA\_ASCII can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), ECDA\_ASCII executes in a DOS box. A run of ECDA\_ASCII is initiated by entering the following command line:

```
ECDA_ASCII
```

Under the Linux operating system ECDA\_ASCII is executed through any of the following Bourne Shell or C Shell commands:

```
ecda_ascii.exe
```

For these commands, “ECDA\_ASCII.EXE” or “ecda\_ascii.exe” is the name of the executable program.

#### **8.1.3 Memory Requirements**

The ECDA\_ASCII code uses dynamic memory allocation. However, only one line of the input file is in memory at any given time, so the memory requirements are minimal. It is expected that most, if not all, of the runs of the ECDA\_ASCII code will require less than 2 MB of memory.

### **8.2 File Definitions**

The ECDA\_ASCII code reads one file and writes one file.

#### **8.2.1 Input Files**

The ECDA\_ASCII code reads an environmental concentration data accumulator (ECDA) file. This file is in binary format and is created by the ECDA program. Other programs such as AIRDROP, GWDROP, CRDROP, RIPSAC, and SOIL write concentration data to these ECDA files.

## 8.2.2 Output Files

The ECDA\_ASCII code writes one data file. This file is a text translation of the binary ECDA file. An example of the first 20 lines of the output text file when only the groundwater media is selected is provided in Table 8.1. The numbers in the left column are record numbers that are not in the original data set. The file starts with an analyte identifier followed by the units of the 11 possible media. The data for multiple realizations are written on each line. This file only contained one realization of data.

**Table 8.1** Excerpts from an ECDA\_ASCII Output File

File: N:\CA1_median\ecda\Eu152_CA1_median.dat				
1	Eu152			
2	Ci/m <sup>3</sup>			
3	Ci/m <sup>3</sup>			
4	Ci/m <sup>3</sup>			
5	Ci/m <sup>3</sup>			
6	Ci/kg			
7	Ci/kg			
8	Ci/kg			
9	Ci/kg			
10	Ci/kg			
11	Ci/m <sup>3</sup>			
12	Ci/m <sup>2</sup> /yr			
13	1945	UH0001	GWAT	0.00000E+00
19	1945	UH0002	GWAT	0.00000E+00
25	1945	UH0003	GWAT	0.00000E+00
31	1945	UH0004	GWAT	0.00000E+00
37	1945	UH0005	GWAT	0.00000E+00
43	1945	UH0006	GWAT	0.00000E+00
49	1945	UH0007	GWAT	0.00000E+00

## 8.3 Code Execution

The ECDA\_ASCII program runs in an interactive mode. A screen image from a run of the code on a Windows machine is provided in Figure 8.1. The user must provide answers to five prompts:

1. The name of the binary ECDA file must be provided. A full pathname is required unless the ECDA file is in the same directory where the ECDA\_ASCII code was invoked.
2. The name of the output file must be provided. The user can select any name. An extension of “.txt” is recommended.
3. The media to translate must be provided (either ALL or one of the 4-character abbreviations provided below). The entry ALL will translate all media. The individual media options are
  - GWAT – concentrations in groundwater (Ci/m<sup>3</sup> or kg/m<sup>3</sup>)
  - SEEP – concentrations in seep water (Ci/m<sup>3</sup> or kg/m<sup>3</sup>)
  - SWAT – concentrations in surface water (river) (Ci/m<sup>3</sup> or kg/m<sup>3</sup>)
  - PWAT – concentrations in river bottom pore water (Ci/m<sup>3</sup> or kg/m<sup>3</sup>)
  - SEDI – concentrations in river bottom sediment (Ci/kg<sub>sediment</sub> or kg<sub>analyte</sub>/kg<sub>sediment</sub>)
  - SORP – concentrations in riparian zone soil (land surface) (Ci/kg<sub>soil</sub> or kg<sub>analyte</sub>/kg<sub>soil</sub>)

- SODR – concentrations in upland soil (land surface) with no irrigation ( $\text{Ci/kg}_{\text{soil}}$  or  $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
  - SOGW – concentrations in upland soil (land surface) with groundwater irrigation ( $\text{Ci/kg}_{\text{soil}}$  or  $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
  - SOSW – concentrations in upland soil (land surface) with surface water irrigation ( $\text{Ci/kg}_{\text{soil}}$  or  $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
  - AIRC – concentrations in air ( $\text{Ci/m}^3$  or  $\text{kg/m}^3$ )
  - AIRD – air deposition rates ( $\text{Ci/m}^2/\text{yr}$  or  $\text{kg/m}^2/\text{yr}$ ).
4. The number of realizations for the simulation must be provided.
  5. The number of data records to be translated must be provided. Care must be taken when translating a large ECDA file such that the output file does not exceed 2.1 gigabytes in size. If the output file exceeds that size, the program will error terminate.

```
D:\SAC\Codes_1>\SAC\CODES_1\ECDA_ASCII\ECDA_ASCII
      Binary to ASCII file conversion program
      SAC Rev. 1 ECDA Concentration files
-----
Program ECDA_ASCII  Version 2.00.1  Dated 07/17/2003
Enter the name of the binary file >
N:\CA1_median\ecda\Eu152_CA1_median.dat
Enter the name of the output file >
Eu152_CA1_median.txt
Enter the media to tranlate ALL for all >
GWAT
Enter the number of realizations >
1
Enter the number of data records to translate >
25000
D:\SAC\Codes_1>
```

**Figure 8.1** Screen Image of an ECDA\_ASCII Run





## **9.0 ECDA\_BACK – Environmental Concentration Fill-In**

### **9.1 Overview**

The ECDA\_BACK code reads a binary environmental concentration data accumulator (ECDA) file, identifies a target set of data for each physical location, and writes the target data to the same physical location for all succeeding time steps. The nominal purpose of this code is to fill in background data for later times for an ECDA file that was only partially filled from a run of the river code (MASS2). The ECDA\_BACK code can only process data for one analyte in a single run of the code.

#### **9.1.1 Location in the Processing Sequence**

The ECDA\_BACK code must be preceded by a run of the ECDA code. In addition, the river code (MASS2) should have been executed to calculate the values used to propagate through the data file.

#### **9.1.2 How the Code Is Invoked**

ECDA\_BACK can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), ECDA\_BACK executes in a DOS box. A run of ECDA\_BACK is initiated by entering the following command line:

```
ECDA_BACK "Keyfilename"
```

Under the Linux operating system ECDA\_BACK is executed through any of the following Bourne Shell or C Shell commands:

```
Ecda_back-1.exe "Keyfilename"
```

For these commands, “ECDA\_BACK.EXE” or “ecda\_back-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If ECDA\_BACK is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If ECDA\_BACK cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **9.1.3 Memory Requirements**

The ECDA program has minimal memory requirements. A production run of the code with an ECDA file containing over 2500 locations required less than 4 MB of memory.

### **9.2 File Definitions**

The ECDA\_BACK program reads four input files and writes to two output files. These files are described in the following sections.

## 9.2.1 Input Files for ECDA\_BACK

### 9.2.1.1 ECBA\_BACK Keyword File

The ECDA\_BACK keyword file contains control information and also defines the concentration values to be inserted in the ECDA files. An example file is provided in Table 9.1. Detailed definitions of the keywords are provided in Section 9.3.

**Table 9.1** Example Keyword File for ECDA\_BACK

```
! Example keywords for ECDA_BACK
REPORT "ECDA_Back_H3.Rpt"
TITLE "ECDA_Back code"
USER "Paul W. Eslinger"
!
FILE ESD "/home/ANALYSIS5/CA_Ref_A/ESD_CA_Ref_A.key"
VERBOSE
!
ANALYTE ID="H3"
MEDIA SWAT PWAT SEDI !SORP AIRC AIRD GWAT SEEP SODR SOGW SOSW
YEAR 2380 ! Hanford signature is gone, natural sources remain
!
END
```

### 9.2.1.2 Other Input Files

The ECDA\_BACK program also reads three other input files. These files are the following:

- ESD keyword file: This file contains the information to locate other input and output files.
- ECDA index file: This file contains the indexing information needed to locate specific data within a binary ECDA file.
- ECDA data file: This file contains the environmental media data for a single analyte and is used for both input and output.

## 9.2.2 Output Files for ECDA\_BACK

The code writes to two output files. These files are a report file and the ECDA concentration file for the selected analyte.

### 9.2.2.1 ECDA\_BACK Report File

The report file contains summary information from the run of the ECDA\_BACK code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the ECDA\_BACK code.

### 9.2.2.2 Concentration File

The ECDA\_BACK program modifies some data in the ECDA file for the analyte identified on the ANALYTE keyword in the ECDA\_BACK keyword file.

## 9.3 Keyword Definitions for ECDA\_BACK

In general, the keywords for the ECDA\_BACK code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 9.3.1 ANALYTE Keyword for ECDA\_BACK

The ANALYTE keyword is used to define the single analyte for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"]
```

The quote string must contain a single analyte ID from the set of the analytes defined in the ESD keyword file. An example of this keyword is the following:

```
ANALYTE "C14"
```

### 9.3.2 END Keyword for ECDA\_BACK

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 9.3.3 FILE Keyword for ECDA\_BACK

The FILE keyword is used to enter the name of the ESD keyword file. The following is this keyword's syntax:

```
FILE [ESD="quote 1"]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. An example of the use of this keyword follows:

```
FILE ESD="/home/ANALYSIS/practice-median/ESD_practice-median.key"
```

#### 9.3.3.1 MEDIA Keyword for ECDA\_BACK

The MEDIA keyword is used to define the medium or media for which data will be processed for a single case. The following is this keyword's syntax:

```
MEDIA [Modifier 1] {Modifier 2} {Modifier 11}
```

The character string is case sensitive and four characters in length. The modifiers associated with the FILE keyword are given in Table 9.2.

**Table 9.2** Modifiers Associated with the MEDIA Keyword in ANIMATE for ECDA Data

Modifier	Description
GWAT	concentrations in groundwater ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEEP	concentrations in seep water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SWAT	concentrations in surface water (river) ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
PWAT	concentrations in river bottom pore water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEDI	concentrations in river bottom sediment ( $\text{Ci}/\text{kg}_{\text{sediment}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{sediment}}$ )
SORP	concentrations in riparian zone soil (land surface) ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SODR	concentrations in upland soil (land surface) with no irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOGW	concentrations in upland soil (land surface) with groundwater irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOSW	concentrations in upland soil (land surface) with surface water irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
AIRC	concentrations in air ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
AIRD	air deposition rates ( $\text{Ci}/\text{m}^2/\text{yr}$ or $\text{kg}/\text{m}^2/\text{yr}$ )

An example of this keyword that modified concentrations in surface water, pore water, and river sediment is the following:

```
MEDIA SWAT PWAT SEDI
```

### 9.3.4 REPORT Keyword for ECDA\_BACK

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "G:\SAC\Sys\EC\Test.rpt"
```

### 9.3.5 TITLE Keyword for ECDA\_BACK

The TITLE keyword is used to define a single-line problem title. The problem title will be written to the report file. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the ECDA_BACK code."
```

### **9.3.6 USER Keyword for ECDA\_BACK**

The USER keyword is used to identify the user of the program. The user name will be written to the report file. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### **9.3.7 VERBOSE Keyword for ECDA\_BACK**

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

### **9.3.8 YEAR Keyword for ECDA\_BACK**

The YEAR keyword is used to identify the single year for which data will be extracted and copied to later years. The program will error terminate if the year is not supplied. The following is this keyword's syntax:

```
YEAR [number]
```

The year is entered as an integer calendar year and should be in the range of years identified in the environmental settings definition keyword file. The following example defines a code run that uses data for the year 2050 and copies it to all later years:

```
YEAR 2050
```



## **10.0 FCDA\_ASCII – Food Concentration Data File Conversions**

### **10.1 Overview**

The FCDA\_ASCII program converts binary food concentration files into text format. The sole purpose of the conversion is to create a file that is human readable in a text editor.

#### **10.1.1 Location in the Processing Sequence**

The FCDA\_ASCII can be executed anytime after the ECEM code has created food concentration data files.

#### **10.1.2 How the Code Is Invoked**

FCDA\_ASCII can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), ECDA\_ASCII executes in a DOS box. A run of FCDA\_ASCII is initiated by entering the following command line:

```
FCDA_ASCII
```

Under the Linux operating system FCDA\_ASCII is executed through any of the following Bourne Shell or C Shell commands:

```
fcda_ascii.exe
```

For these commands, “FCDA\_ASCII .EXE” or “fcda\_ascii.exe” is the name of the executable program.

#### **10.1.3 Memory Requirements**

The FCDA\_ASCII code uses dynamic memory allocation. However, only one line of the input file is in memory at any given time, so the memory requirements are minimal. It is expected that most, if not all, of the runs of the FCDA\_ASCII code will require less than 2 MB of memory.

### **10.2 File Definitions**

The FCDA\_ASCII code reads one file and writes one file.

#### **10.2.1 Input Files**

The FCDA\_ASCII code reads a food concentration data accumulator (FCDA) file. This file is in binary format and is created by the ECEM program.

#### **10.2.2 Output Files**

The FCDA\_ASCII code writes one data file. This file is a text translation of the binary FCDA file. An example of the first 20 lines of the output text file for a file is provided in Table 10.1. The numbers in the left column are record numbers that are not in the original data set. The file starts with an analyte identifier followed by species identifier and the location type identifier. This file only contained one realization of data.

**Table 10.1** Excerpts from an FCDA\_ASCII Output File

File: N:\CA1_median\foods\Food_C14_UUPIGS.fod			
1	C14		
2	UUPIGS		
3	UPLAND		
4	1945	UH0001	1.85791E+00
5	1945	UH0001	1.85791E+00
6	1945	UH0001	1.85791E+00
7	1945	UH0002	1.85791E+00
8	1945	UH0002	1.85791E+00
9	1945	UH0002	1.85791E+00
10	1945	UH0003	1.85791E+00
11	1945	UH0003	1.85791E+00
12	1945	UH0003	1.85791E+00
13	1945	UH0004	1.39627E+00
14	1945	UH0004	1.39627E+00
15	1945	UH0004	1.39627E+00
16	1945	UH0005	1.39627E+00
17	1945	UH0005	1.39627E+00
18	1945	UH0005	1.39627E+00
19	1945	UH0006	1.85791E+00

### 10.3 Code Execution

The FCDA\_ASCII program runs in an interactive mode. A screen image from a run of the code on a Windows machine is provided in Figure 10.1. The user must provide answers to four prompts:

1. The name of the binary FCDA file must be provided. A full pathname is required unless the FCDA file is in the same directory where the FCDA\_ASCII code was invoked.
2. The number of realizations for the simulation must be provided.
3. The name of the output file must be provided. The user can select any name. An extension of “.txt” is recommended.
4. The number of data records to be translated must be provided. Care must be taken when translating a large FCDA file such that the output file does not exceed 2.1 gigabytes in size. If the output file exceeds that size, the program will error terminate.

```

D:\SAC\Codes_1>\SAC\CODES_1\FCDA_ASCII\FCDA_ASCII

      Binary to ASCII file conversion program
      SAC Rev. 1 FCDA Concentration files
-----
Program FCDA_ASCII  Version 1.00.1  Dated 03/12/2003

Enter the name of the binary file >
N:\CA1_median\foods\Food_C14_UUPIGS.fod
Enter the number of realizations >
1
Enter the name of the output file >
Food_C14_UUPIGS.txt
Enter the number of data records to translate >
125
D:\SAC\Codes_1>_

```

**Figure 10.1** Screen Image of an FCDA\_ASCII Run



## **11.0 FILLECDA – Environmental Concentration Data File Modifications**

### **11.1 Overview**

The ECDA program creates environmental concentration data accumulator (ECDA) files and initializes the contents. Other programs (GWDROP, CRDROP, AIRDROP, RIPSAC, and SOIL) then fill in the concentration values for different media. The FILLECDA program provides the capability to insert user-specified concentration values into any record in an ECDA file.

#### **11.1.1 Location in the Processing Sequence**

The FILLECDA program can be used anytime after the ECDA files are created by the ECDA program.

#### **11.1.2 How the Code Is Invoked**

FILLECDA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), FILLECDA executes in a DOS box. A run of FILLECDA is initiated by entering the following command line:

```
FILLECDA "Keyfilename"
```

Under the Linux operating system FILLECDA is executed through any of the following Bourne Shell or C Shell commands:

```
filelecda-1.exe "Keyfilename"
```

For these commands, "FILLECDA.EXE" or "filelecda-1.exe" is the name of the executable program, and "Keyfilename" is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If FILLECDA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If FILLECDA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **11.1.3 Memory Requirements**

The FILLECDA program has minimal memory requirements. The memory is small because data for only a single record of an ECDA file is generated and written at any one time.

### **11.2 File Definitions**

The FILLECDA program reads three input files and writes to two or more output files. These files are described in the following paragraphs.

## 11.2.1 Input Files

### 11.2.1.1 FILLECDA Keyword File

The FILLECDA keyword file contains control information and also defines the concentration values to be inserted in the ECDA files. An example file is provided in Table 11.1. Detailed definitions of the keywords are provided in Section 11.3.

**Table 11.1** Example Keyword File for the FILLECDA Program

```
! Keyword file for the Fillecda program
REPORT "Test.Rpt"           ! Report file name (first keyword)
TITLE "Test the Fillecda Code" ! Title for labeling purposes
USER "Paul W. Eslinger"      ! User name for labeling purposes
FILE ESD "ESD.Key"          ! Name of the ESD keyword file
REALIZAT 1                   ! Number of realizations
DATA LOCATION="EL0001" ANALYTE="Cl4" YEAR=10000 MEDIA="GWAT" VALUES=1
DATA LOCATION="EL0001" ANALYTE="I129" YEAR=10000 MEDIA="SEEP" VALUES=1.1
DATA LOCATION="EL0001" ANALYTE="U236" YEAR=10000 MEDIA="SORP" VALUES=1.2
DATA LOCATION="EL0002" ANALYTE="U238" YEAR=10000 MEDIA="SWAT" VALUES=11
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="SWAT" VALUES=11.1
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="PWAT" VALUES=11.2
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="SDAT" VALUES=11.3
DATA LOCATION="EL0003" ANALYTE="Tc99" YEAR=10000 MEDIA="AIRC" VALUES=11.4
DATA LOCATION="EL0003" ANALYTE="Tc99" YEAR=10000 MEDIA="AIRD" VALUES=11.5
END
```

### 11.2.1.2 Other Input Files

The FILLECDA program also reads two other input files. These files are the ESD keyword file and the ECDA index file. The ESD keyword file contains the names of the concentration files that are to be modified. The ECDA map file contains index data related to writing to the ECDA concentration files.

## 11.2.2 Output Files

The code writes to two or more output files. These files are a report file and one or more ECDA concentration files.

### 11.2.2.1 FILLECDA Report File

The report file contains summary information from the run of the FILLECDA code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the FILLECDA code.

### 11.2.2.2 Concentration Files

The FILLECDA program writes some data to an ECDA file for every analyte identified on a DATA keyword in the FILLECDA keyword file.

## 11.3 Keyword Definitions for FILLECDA

In general, the keywords for FILLECDA can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 11.3.1 DATA Keyword for FILLECDA

The DATA keyword is used to define the data values to be inserted into one or more ECDA files. The following is this keyword's syntax:

```
DATA [LOCATION="quote"] [MEDIA="quote"] [ANALYTE="quote"] [YEAR=number]
    [VALUES=number, ..., number]
```

This keyword identifies the data for all realizations of concentration for a specific analyte, location, media, and time combination. More than one DATA keyword can be entered. The modifiers for the DATA keyword are defined in Table 11.2.

**Table 11.2** Modifiers Associated with the DATA Keyword in FILLECDA

Modifier	Description
ANALYTE	The quote string associated with the ANALYTE modifier contains the analyte ID for the concentration data values. The analyte ID must be one of the set of analytes defined in the ESD keyword file.
LOCATION	The quote string associated with the LOCATION modifier contains the location ID for the concentration data values. The location ID must be one of the set of locations defined in the ESD keyword file.
MEDIA	The quote string associated with the MEDIA modifier contains the media ID for the concentration data values. The media ID must be one of the following: GWAT, SEEP, SWAT, PWAT, SEDI, SORP, SODR, SOGW, SOSW, AIRC, or AIRD.
YEAR	The numerical value associated with the YEAR modifier identifies the calendar year at which the concentration data apply. The year entered must be one of the years defined on the TIMES keyword in the ESD keyword file.
VALUES	The numerical values associated with the VALUES modifier contain the concentration values to be entered in the ECDA file. There must be as many values entered as there are realizations defined in the ESD keyword file.

If a DATA keyword does not identify a valid analyte, location, media, and time combination in the ECDA data file, then the information on that DATA keyword is discarded and an error message is written to the report file. Example entries for the DATA keyword for a case of two realizations are the following:

```
DATA LOCATION="EL01" ANALYTE="I129" YEAR=10000 MEDIA="SEEP" VALUES=1.1 1.3
DATA LOCATION="EL01" ANALYTE="U236" YEAR=10000 MEDIA="SORP" VALUES=1.2 1.4
DATA LOCATION="EL02" ANALYTE="U238" YEAR=10000 MEDIA="SWAT" VALUES=11 12
DATA LOCATION="EL02" ANALYTE="Tc99" YEAR=10000 MEDIA="SWAT" VALUES=11.1 10
DATA LOCATION="EL02" ANALYTE="Tc99" YEAR=10000 MEDIA="PWAT" VALUES=11.2 12
```

### 11.3.2 END Keyword for FILLECDA

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 11.3.3 FILE Keyword for FILLECDA

The FILE keyword is used to enter the name of the ESD keyword file. The following is this keyword's syntax:

```
FILE [ESD="quote 1"]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. An example of the use of this keyword follows:

```
FILE ESD="/home/ANALYSIS/practice-median/ESD_practice-median.key"
```

### 11.3.4 REALIZAT Keyword for FILLECDA

The REALIZ (or REALIZATION) keyword defines the number of realizations to process. The following is this keyword's syntax:

```
REALIZATION value1
```

This keyword gives the number of realizations of data that will be entered on each DATA record. The number must match exactly with the number of realizations in the ESD keyword file. The following keyword record sets the number of realizations to 10:

```
REALIZAT 10
```

### 11.3.5 REPORT Keyword for FILLECDA

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/SystemCodes/Cultural/Test1.rpt"
```

### 11.3.6 TITLE Keyword for FILLECDA

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the FILLECDA code."
```

### 11.3.7 USER Keyword for FILLECDA

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```



## **12.0 FILLFCDA – Food Concentration Data File Modifications**

### **12.1 Overview**

The ECEM program creates food concentration data accumulator (FCDA) files containing concentrations in food products that are utilized in the HUMAN code. The FILLFCDA program provides the capability to insert user-specified concentration values into any record in an FCDA file.

#### **12.1.1 Location in the Processing Sequence**

The FILLFCDA program can be used anytime after the FCDA files are created by the ECEM program.

#### **12.1.2 How the Code Is Invoked**

FILLFCDA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), FILLFCDA executes in a DOS box. A run of FILLFCDA is initiated by entering the following command line:

```
FILLFCDA "Keyfilename"
```

Under the Linux operating system FILLFCDA is executed through any of the following Bourne Shell or C Shell commands:

```
fillfcda-1.exe "Keyfilename"
```

For these commands, “FILLFCDA.EXE” or “fillfcda-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If FILLFCDA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If FILLFCDA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **12.1.3 Memory Requirements**

The FILLFCDA program has moderate memory requirements. The major memory driver is storing the information from all DATA keywords before any is processed. If memory requirements become excessive, the problem can be subdivided into multiple runs, each using a subset of the data.

### **12.2 File Definitions**

The FILLFCDA program reads three input files and writes to two or more output files. These files are described in the following paragraphs.

## 12.2.1 Input Files

### 12.2.1.1 FILLFCDA Keyword File

The FILLFCDA keyword file contains control information and also defines the concentration values to be inserted in the FCDA files. An example file is provided in Table 12.1. Detailed definitions of the keywords are provided in Section 12.3.

**Table 12.1** Example Keyword File for the FILLFCDA Program

```
! Keyword file for the FillFCDA program
REPORT "Test.Rpt"           ! Report file name (first keyword)
TITLE "Test the FillFCDA Code" ! Title for labeling purposes
USER "Paul W. Eslinger"      ! User name for labeling purposes
FILE ESD "ESD.Key"           ! Name of the ESD keyword file
REALIZAT 1                   ! Number of realizations
! Upland locations
DATA LOCATION="UH0002" ANALYTE="H3" YEAR=2150 SOIL="SOSW"
  SPECIES="UCATMT" VALUES=0.83482
DATA LOCATION="UH0002" ANALYTE="H3" YEAR=2150 SOIL="SOGW"
  SPECIES="UCATMT" VALUES=0.93036
DATA LOCATION="UH0002" ANALYTE="H3" YEAR=2150 SOIL="SODR"
  SPECIES="UCATMT" VALUES=0.98167
!
! Riparian Locations
DATA LOCATION="RHP001" ANALYTE="H3" YEAR=2100 SPECIES="RMLBRY"
  VALUES=0.63039
DATA LOCATION="RHP001" ANALYTE="H3" YEAR=2100 SPECIES="RMDEER"
  VALUES=0.01514
DATA LOCATION="RHP001" ANALYTE="H3" YEAR=2100 SPECIES="RMALRD"
  VALUES=0.76302
!
! Aquatic locations
DATA LOCATION="QHP001" ANALYTE="H3" YEAR=2100 SPECIES="QCARPS"
  VALUES=0.61637
DATA LOCATION="QHP001" ANALYTE="H3" YEAR=2100 SPECIES="QSBASS"
  VALUES=0.29431
DATA LOCATION="QHP001" ANALYTE="H3" YEAR=2100 SPECIES="QMWTF5"
  VALUES=0.86436
!
END
```

### 12.2.1.2 Other Input Files

The FILLFCDA program also reads two other input files. These files are the ESD keyword file and the FCDA index file. The ESD keyword file contains the names of the concentration files that are to be modified. The FCDA map file contains index data related to writing to the FCDA concentration files.

## 12.2.2 Output Files

The code writes to two or more output files. These files are a report file and one or more FCDA concentration files.



### 12.2.2.1 FILLFCDA Report File

The report file contains summary information from the run of the FILLFCDA code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the FILLFCDA code.

### 12.2.2.2 Concentration Files

The FILLFCDA program writes some data to an FCDA file for every combination of analyte and SPECIES identified on DATA keywords in the FILLFCDA keyword file.

## 12.3 Keyword Definitions for FILLFCDA

In general, the keywords for FILLFCDA can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 12.3.1 DATA Keyword for FILLFCDA

The DATA keyword is used to define the data values to be inserted into one or more FCDA files. The following is this keyword's syntax:

```
DATA [LOCATION="quote"] [SPECIES="quote"] [ANALYTE="quote"] [YEAR=number]
    {SOIL="quote"} [VALUES=number, ..., number]
```

This keyword identifies the data for all realizations of concentration for a specific analyte, location, soil type, species, and time combination. More than one DATA keyword can be entered. The modifiers for the DATA keyword are defined in Table 12.2.

**Table 12.2** Modifiers Associated with the DATA Keyword in FILLFCDA

Modifier	Description
ANALYTE	The quote string associated with the ANALYTE modifier contains the analyte ID for the concentration data values. The analyte ID must be one of the set of analytes defined in the ESD keyword file.
LOCATION	The quote string associated with the LOCATION modifier contains the location ID for the concentration data values. The location ID must be one of the set of locations defined in the ESD keyword file.
SOIL	The quote string associated with the optional SOIL modifier contains the ID for a soil type. This modifier is not used if an aquatic or riparian species is specified. If a terrestrial species living in the upland zone is defined, then one of the values “SODR”, “SOGW”, or “SOSW” must be used. These values define, respectively, soil with no irrigation, soil with groundwater used for irrigation, and soil with surface water used for irrigation.

Modifier	Description
SPECIES	The quote string associated with the SPECIES modifier contains the ID for an ecological species. The species ID must be one of the ecological species defined in the ESD keyword file.
YEAR	The numerical value associated with the YEAR modifier identifies the calendar year at which the concentration data apply. The year entered must be one of the years defined on the TIMES keyword in the ESD keyword file.
VALUES	The numerical values associated with the VALUES modifier contain the concentration values to be entered in the FCDA file. As many values must be entered as there are realizations defined in the ESD keyword file.

If a DATA keyword does not identify a valid analyte, location, species, soil, and time combination in the FCDA data file, then the information on that DATA keyword is discarded and an error message is written to the report file. Example entries for the DATA keyword for a case of one realization are the following:

```

! Upland locations
DATA LOCATION="UH0002" ANALYTE="H3" YEAR=2150 SOIL="SOSW"
SPECIES="UCATMT" VALUES=0.83482
DATA LOCATION="UH0002" ANALYTE="H3" YEAR=2150 SOIL="SOGW"
SPECIES="UCATMT" VALUES=0.93036
!
! Riparian Locations
DATA LOCATION="RHP001" ANALYTE="H3" YEAR=2100 SPECIES="RMLBRY"
VALUES=0.63039
DATA LOCATION="RHP001" ANALYTE="H3" YEAR=2100 SPECIES="RMDEER"
VALUES=0.01514
!
! Aquatic locations
DATA LOCATION="QHP001" ANALYTE="H3" YEAR=2100 SPECIES="QCARPS"
VALUES=0.61637
DATA LOCATION="QHP001" ANALYTE="H3" YEAR=2100 SPECIES="QSBASS"
VALUES=0.29431

```

### 12.3.2 END Keyword for FILLFCDA

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 12.3.3 FILE Keyword for FILLFCDA

The FILE keyword is used to enter the name of the ESD keyword file. The following is this keyword's syntax:

```
FILE [ESD="quote 1"]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. An example of the use of this keyword follows:

```
FILE ESD="/home/ANALYSIS/practice-median/ESD_practice-median.key"
```

### 12.3.4 REALIZAT Keyword for FILLFCDA

The REALIZ (or REALIZATION) keyword defines the number of realizations to process. The following is this keyword's syntax:

```
REALIZAT value1
```

This keyword gives the number of realizations of data that will be entered on each DATA record. The number must match exactly with the number of realizations in the ESD keyword file. The following keyword record sets the number of realizations to 10:

```
REALIZAT 10
```

### 12.3.5 REPORT Keyword for FILLFCDA

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/SystemCodes/Cultural/Test1.rpt"
```

### 12.3.6 TITLE Keyword for FILLFCDA

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the FILLFCDA code."
```

### 12.3.7 USER Keyword for FILLFCDA

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

## **13.0 GETTRACE – Human Risk Extractor**

### **13.1 Overview**

The GETTRACE program gathers up a suite of results by analyte after the HIGHIMPACT code has identified the location by year where the maximum dose occurs (deterministic runs only).

#### **13.1.1 Location in the Processing Sequence**

The GETTRACE code must follow a run of the HIGHIMPACT code, which is a post-processor of HUMAN code results.

#### **13.1.2 How the Code Is Invoked**

GETTRACE can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), GETTRACE executes in a DOS box. A run of GETTRACE is initiated by entering the following command line:

```
GETTRACE "Keyfilename"
```

Under the Linux operating system GETTRACE is executed through any of the following Bourne Shell or C Shell commands:

```
gettrace-1.exe "Keyfilename"
```

For these commands, “GETTRACE.EXE” or “gettrace-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If GETTRACE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If GETTRACE cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **13.1.3 Memory Requirements**

The GETTRACE program has minimal memory requirements. A run of the code to extract values for 2400 upland locations required less than 2 MB of memory.

### **13.2 File Definitions**

The GETTRACE program reads three input files and writes two output files. These files are described in the following sections.

#### **13.2.1 Input Files**

The GETTRACE program reads a control keyword file, a results file from a run of the HUMAN code, and a results file from the HIGHIMPACT code containing the information about the highest impact

calculated by the HUMAN code over a set of locations for every year (see Table 16.2). The GETTRACE keyword file contains control information. An example file is provided in Table 13.1. Detailed definitions of the keywords are provided in Section 13.3.

**Table 13.1** Example Keyword File for the GETTRACE Program

```
! Keyword file for GETTRACE
REPORT "Trace_History2_Best_ResFarmer.Rpt"
USER "Paul W. Eslinger"
!
FILE VALUE "Human_History2_Best_ResFarmer_Dtl.csv"
FILE TRACE "High_History2_SOrig_ResFarmer.csv"
FILE OUTPUT "Trace_History2_Best_ResFarmer.csv"
HEADER OUTPUT
!
! Change dose from rem to millirem (Factor of 1000.0)
FACTOR 1000.0
!
! Select the set of outputs
ANALYTE "COMBIN","RAD","SUMDOSE"
ANALYTE "H3","RAD","ANADOSE"
ANALYTE "Tc99","RAD","ANADOSE"
ANALYTE "I129","RAD","ANADOSE"
!
END
```

### 13.2.2 Output Files

The GETTRACE program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the output file containing the information about the highest impacts for the maximum location for every year. An example results file provided in Table 13.2 gives the output associated with the keyword file provided in Table 13.1.

**Table 13.2** Example Output File from the GETTRACE Code

"YEAR"	"COMBIN"	"H3"	"Tc99"	"I129"
1945	0.00000E+00	3.87605E-02	2.24642E-02	7.41619E-03
1950	0.00000E+00	8.70292E-02	7.72445E-01	1.20024E-01
1955	0.00000E+00	1.49553E+03	2.74575E+00	6.05045E-02
1960	0.00000E+00	9.40765E+02	1.57715E-01	9.44364E+00
1965	0.00000E+00	3.86130E+03	6.06795E-01	1.41852E+02
1970	0.00000E+00	4.59619E+03	1.58208E+00	2.98593E+02
1975	0.00000E+00	3.17458E+03	1.76993E+00	2.91658E+02
1980	0.00000E+00	1.72096E+03	1.32152E+00	2.53054E+02
1985	0.00000E+00	9.60606E+02	8.16556E-01	1.98844E+02
1990	0.00000E+00	5.34215E+02	5.05104E-01	1.28364E+02
1991	0.00000E+00	4.70447E+02	4.81876E-01	1.22545E+02
1992	0.00000E+00	4.09161E+02	4.60910E-01	1.15321E+02
1993	0.00000E+00	3.43649E+02	4.29532E-01	1.06211E+02
1994	0.00000E+00	2.85349E+02	3.93153E-01	9.76629E+01
1995	0.00000E+00	2.43559E+02	2.92475E-01	1.09639E+02

1996,	0.00000E+00,	2.24035E+02,	2.89320E-01,	1.07062E+02
1997,	0.00000E+00,	2.05260E+02,	2.84110E-01,	1.04391E+02
1998,	0.00000E+00,	1.88809E+02,	2.79285E-01,	9.77042E+01
1999,	0.00000E+00,	1.73436E+02,	2.73704E-01,	9.55426E+01
2000,	0.00000E+00,	1.58697E+02,	2.67182E-01,	9.33077E+01
2001,	0.00000E+00,	1.44738E+02,	2.59829E-01,	9.10630E+01
2002,	0.00000E+00,	1.31631E+02,	2.51869E-01,	8.88344E+01
2003,	0.00000E+00,	1.19977E+02,	2.44429E-01,	8.68767E+01
2004,	0.00000E+00,	1.09206E+02,	2.36763E-01,	8.49621E+01
2005,	0.00000E+00,	1.01727E+02,	2.20914E-01,	8.50020E+01
2006,	0.00000E+00,	9.46203E+01,	2.31884E-01,	8.42241E+01
2007,	0.00000E+00,	8.77932E+01,	2.30197E-01,	8.32751E+01
2008,	0.00000E+00,	8.13146E+01,	2.26671E-01,	8.22686E+01
2009,	0.00000E+00,	7.52102E+01,	2.22700E-01,	8.12434E+01
2010,	0.00000E+00,	6.94410E+01,	2.18370E-01,	8.01860E+01
2011,	0.00000E+00,	6.39005E+01,	2.13480E-01,	7.90383E+01
2012,	0.00000E+00,	5.87525E+01,	2.08503E-01,	7.79072E+01
2013,	0.00000E+00,	5.40766E+01,	2.03776E-01,	7.68563E+01
2014,	0.00000E+00,	4.96809E+01,	1.98801E-01,	7.57698E+01
2015,	0.00000E+00,	4.55116E+01,	1.93475E-01,	7.46234E+01
2016,	0.00000E+00,	4.16644E+01,	1.88171E-01,	7.34939E+01
2017,	0.00000E+00,	5.93251E+01,	2.58107E-01,	4.58954E+01
2018,	0.00000E+00,	5.55198E+01,	2.55514E-01,	4.55337E+01
2019,	0.00000E+00,	5.19433E+01,	2.52859E-01,	4.51847E+01
2020,	0.00000E+00,	4.85760E+01,	2.50108E-01,	4.48265E+01

### 13.3 Keyword Definitions for the GETTRACE Code

In general, the keywords for the GETTRACE code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

#### 13.3.1 ANALYTE Keyword for GETTRACE

The ANALYTE keyword is used to define the analytes and solutions for which data will be processed. The data to be processed are obtained from a “details” file output by the HUMAN code. Examination of the file to be processed can aid in selecting options for the ANALYTE keyword. More than one ANALYTE keyword can be used in a single run of the GETTRACE code. The following is this keyword’s syntax:

```
ANALYTE ["quote 1" "quote 2" "quote 3"]
```

The first quote string must be an analyte ID from the set of analytes in the ESD keyword file, or the string “COMBIN” if the combined dose over multiple analytes is desired.

The second quote string is the analyte type. The analyte type quote string is case sensitive and eight characters in length. The modifiers associated with the analyte type for the ANALYTE keyword are given in Table 13.3.

**Table 13.3** Modifiers Associated with the Analyte Type on the ANALYTE Keyword in GETTRACE

Modifier	Description
ALL	The analyte field is used for a quantity that is a sum over analytes.
CAR	The analyte is a carcinogenic chemical.
CONMEDIA	The analyte field is used to select an environmental medium.
CONFOODS	The analyte field is used to select a food species.
HAZ	The analyte is a hazardous chemical.
RAD	The analyte is a radionuclide.

The third quote string is the solution type output by the HUMAN code. The solution type quote string is case sensitive and is two to seven characters in length. The modifiers associated with the solution type for the ANALYTE keyword are given in Table 13.4.

**Table 13.4** Modifiers Associated with the Solution Type on the ANALYTE Keyword in GETTRACE

Modifier	Description
<b>Solution Type Modifiers Associated with Analyte Type CONMEDIA</b>	
AIRC	Air concentrations
GWAT	Groundwater concentrations
SEDI	Sediment concentrations
SEEP	Seep water concentrations
SODR	Soil concentrations, no irrigation
SOGW	Soil concentrations, groundwater irrigated
SORP	Riparian soil concentrations
SOSW	Soil concentrations, surface water irrigated
SWAT	Surface water concentrations
<b>Solution Type Modifiers Associated with Analyte Type CONFOODS</b>	
BIRD	Food concentration, birds
EGGS	Food concentration, eggs
FISH	Food concentration, fish
FISH_2	Food concentration, 2 <sup>nd</sup> fish type
FISH_3	Food concentration, 3 <sup>rd</sup> fish type
FRUIT	Food concentration, fruit
GRAIN	Food concentration, grain
LEAFVEG	Food concentration, leafy vegetables
MEAT	Food concentration, meat
MILK	Food concentration, milk
ROOTVEG	Food concentration, root vegetables



<b>Solution Type Modifiers Associated with Analyte Types CAR, HAZ, RAD</b>	
ANADOSE	Analyte dose
ANARISK	Analyte risk (CAR and RAD)
ANAHQ	Analyte hazard quotient (HAZ)
DOSEING	Ingestion dose (CAR, HAZ and RAD)
DOSEINH	Inhalation dose (CAR, HAZ and RAD)
DOSEEXT	External dose (RAD)
DOSEDER	Dermal dose (CAR and HAZ)
HQING	Ingestion hazard quotient (HAZ)
HQINH	Inhalation hazard quotient (HAZ)
HQDER	Dermal hazard quotient (HAZ)
POPDOSE	Population dose (RAD)
POPRISK	Population risk (RAD)
RISKING	Ingestion risk (CAR and RAD)
RISKINH	Inhalation risk (CAR and RAD)
RISKEXT	External risk (RAD)
RISKDER	Dermal risk (CAR)
SUMDOSE	Dose summed over analytes (CAR and HAZ)
SUMHQ	Hazard quotient summed over analytes (HAZ)
<b>Solution Type Modifiers Associated with Analyte Type ALL</b>	
SUMRISK	Risk summed over carcinogenic chemicals and radionuclides

Two examples of this keyword are the following:

```
ANALYTE "COMBIN" , "ALL" , "SUMDOSE"
ANALYTE "H3" , "RAD" , "ANADOSE"
```

### 13.3.2 END Keyword for GETTRACE

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 13.3.3 FACTOR Keyword for GETTRACE

The FACTOR keyword is used to enter a multiplicative factor to change the units of the data for output. Use a value of 1 if the input units are to be preserved. The following is this keyword's syntax:

```
FACTOR value1
```

An example keyword that converts the radionuclide dose units from rem to mrem upon output is the following:

```
FACTOR 1000.0
```

### 13.3.4 FILE Keyword for GETTRACE

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"} {modifier3="quote3"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the input file defining human results is associated with the modifier VALUE. The name of the output file from HIGHIMPACT is associated with the modifier TRACE. The output file to be written is associated with the modifier OUTPUT. Example file keywords that define these three files are the following:

```
FILE VALUE  "RipNativeAmer_CAL_median_Dtl.csv"  
FILE TRACE  "RipNativeAmer_CAL_median_High.csv"  
FILE OUTPUT "RipNativeAmer_CAL_median_Trace.csv"
```

### 13.3.5 HEADER Keyword for GETTRACE

The HEADER keyword is used to control whether or not a header line is written to the output file. The following is this keyword's syntax:

```
HEADER [OUTPUT]
```

If this keyword is present with the modifier OUPUT, then a header line is written to the output file identified in the FILE OUTPUT keyword. Otherwise, the data are written with no header information.

### 13.3.6 REPORT Keyword for GETTRACE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "RipNativeAmer_CAL_median_Trace.Rpt"
```

### **13.3.7 USER Keyword for GETTRACE**

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```



## 14.0 GWGRAB – Groundwater Transport Module Data Extractor

### 14.1 Overview

The GWGRAB code provides an efficient means for analyzing inputs, outputs, mass balance, and other parameters associated with the Groundwater Transport module of the SAC Rev. 1. GWGRAB can detect the parameters of an assessment, locate all necessary inputs and results in the SAC directory structure, and summarize results in a number of ways that can be imported into spreadsheets or graphing programs.

#### 14.1.1 Location in the Processing Sequence

The GWGRAB reads data written by the groundwater flow and transport module. The GWGRAB code must be run after the CFEST code. GWGRAB can also be used to monitor the progress of CFEST runs.

#### 14.1.2 How the Code Is Invoked

GWGRAB only runs under the Linux operating system. GWGRAB is executed through the following Bourne Shell or C Shell commands:

```
gwgrab-1.exe "ESDKeyfilename" "GWGRABKeyFileName"
```

For the command, “gwgrab-1.exe” is the name of the executable program, and “ESDKeyfilename” is the name of the ESD keyword file providing overall control to the SAC simulation, and “GWGRABKeyFileName” is the name of the GWGRAB control keyword file. If GWGRAB is invoked without two filenames, the code will terminate execution after writing an error message to the standard output device. GWGRAB must be invoked from the root directory of a SAC analysis set, and the ESD keyword file name must be the local file name (no path information). The keyword file name can include path information.

#### 14.1.3 Memory Requirements

The GWGRAB program can require a large amount of memory. Dynamic memory allocation is used, and memory use depends on the number of realizations and number of analytes. A typical run with 13 analytes and 100 realizations required on the order of 65 MB of memory.

### 14.2 File Definitions

The GWGRAB program requires two input files to control data extraction. The program also writes two or more output files for every run.

#### 14.2.1 Input Files

The GWGRAB program requires two input files. The first file is the environmental settings definition (ESD) keyword file providing overall control for the SAC problem. This file contains information used by GWGRAB such as the number of realizations and analyte definitions. The second input file is the GWGRAB control keyword file that gives specific information concerning the particular data to be extracted and reported. This file may be named with any name so that any number of control keyword

files may be created at any one time. However, only one control file may be used per run of the GWGRAB program

### 14.2.2 Output Files

Output files include a log file and up to five additional files. The log file will be named by the same name as the GWGRAB control keyword file with the extension of “.log.” The other five files, if written, contain analyte influx data, analyte storage data, analyte outgoing flux data, decay information, and error messages.

## 14.3 Keyword Descriptions for the GWGRAB Control Keyword File

All user instructions regarding data extraction are defined in the GWGRAB keyword control file. Keywords may be listed in any order. However, all information after the END keyword will be ignored. Therefore, it is necessary that the END keyword be the last keyword listed.

### 14.3.1 ANALYTE Keyword for GWGRAB

The ANALYTE keyword is used to define the analyte or analytes to process. The following is this keyword’s syntax:

```
ANALYTE [ID="quote1" {"quote2"} ... {"quoteN"} |ALL]
```

Several ANALYTE keywords may be entered, or a single ANALYTE keyword with more than one ID can be used. If the modifier ALL is used, then the report is generated for all analytes defined in the ESD keyword file. If the modifier ID is used, it must be followed by one or more quote strings containing analyte IDs. Each analyte ID is limited to six characters in length and must identify one of the analytes defined in the ESD control file. The following set of three analyte keywords have the same effect as the single fourth keyword.

```
ANALYTE ID= "H3"  
ANALYTE ID= "Sr90"  
ANALYTE ID= "Cs137"  
ANALYTE ID= "H3" "Cs137" "Sr90"
```

### 14.3.2 END Keyword for GWGRAB

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword’s syntax:

```
END
```

### 14.3.3 EXEDIR Keyword for GWGRAB

The EXEDIR keyword identifies the directory location for program executable files. The following is this keyword’s syntax:

```
EXEDIR ["quote"]
```

The EXEDIR keyword is used to identify the locations of programs that get executed through calls from GWGRAB. The programs needed are part of the CFEST suite of codes. The following is a sample keyword:

```
EXEDIR "/home/CODES/bin"
```

#### 14.3.4 EXTRACT Keyword for GWGRAB

The EXTRACT keyword defines the types of data to be extracted. Several EXTRACT keywords with different modifiers may be included in one keyword file, but a single EXTRACT keyword with more than one modifier is also acceptable. The following is this keyword's syntax:

```
EXTRACT [INQUIRE | BALANCE | CHECK]
```

The modifiers for the EXTRACT keyword are defined in Table 14.1.

**Table 14.1** Modifiers Associated with the EXTRACT keyword for GWGRAB

Modifier	Description
INQUIRE	Returns information about the SAC run, such as which analytes and how many realizations were run
BALANCE	Calculates the amount of analyte (kg) in the groundwater domain at the SAC-wide balance times (use EXTRACT INQUIRE to determine balance times)
CHECK	Reports the number of completed CFEST time steps for one or more CFEST runs

#### 14.3.5 PATH Keyword for GWGRAB

The PATH keyword defines the path to the directory where the extracted results are stored. The path must be entered in quotes. The following is this keyword's syntax:

```
PATH "path name"
```

An example PATH keyword record is the following:

```
PATH "/home/ANALYSIS2/Initial3/gwgrab/"
```

#### 14.3.6 REALIZAT Keyword for GWGRAB

The REALIZATION (or REALIZATION) keyword defines which realization or realizations to report. Several REALIZAT keywords may be entered, or a single REALIZAT keyword with more than one numbers may be used. Realization numbers must be entered as positive integers. The following is this keyword's syntax:

```
REALIZATION [N1 {N2} ... {Nk} | ALL]
```

If the ALL modifier is entered, then data will be extracted for all realizations defined in the ESD keyword file. If the ALL modifier is absent, then data will be extracted for only the specified realizations. The following two keywords select extractions for realizations 1, 2, 3, 10, 27, and 45

```
REALIZAT 45 2 3
```

REALIZAT 1 10 27

### 14.3.7 OUTPUT Keyword for GWGRAB

The OUTPUT keyword defines how to format the output data. The following is this keyword's syntax:

OUTPUT [RAW]

The RAW modifier must be present. Raw data are output on the time steps used in CFEST. An example use of this keyword is the following:

OUTPUT RAW

### 14.3.8 TIME Keyword for GWGRAB

The TIME Keyword defines how to present the data with respect to time. The following is this keyword's syntax:

TIME [ANNUAL|TOTAL]

If the modifier ANNUAL is used, then data are summed and registered to calendar years. If the modifier TOTAL is used, then data are summed for the entire simulation period. Two examples of this keyword record are the following:

TIME ANNUAL  
TIME TOTAL

## 14.4 Example Cases for GWGRAB

Two examples uses of GWGRAB are provided in the following sections.

### 14.4.1 GWGRAB Example – Check Run Status for CFEST

GWGRAB can be used to check the status of the Groundwater Transport (CFEST) module during an active SAC assessment. This task is accomplished by use of the EXTRACT CHECK keyword in the GWGRAB keyword control file. An example of the GWGRAB keyword control file is provided in Table 14.2.

**Table 14.2** Example Keyword File for the GWGRAB Program - Check Run Status

PATH	"/home/ANALYSIS2/Initial3/gwgrab/"	!Where to put the results
EXEDIR	"/home/CODES/bin/"	!Where the SAC executables are
EXTRACT CHECK		!Report completed CFEST time steps.
ANALYTE ALL		!Report all analytes.
REALIZATION ALL		!Report all realizations.
TIME ANNUAL		!TIME keyword is not applicable but must be !included
OUTPUT RAW		!Report raw data for each realization.
END		



The keyword control in Table 14.2 results in the following type of output. Only an excerpt from the output is provided in Table 14.3 in order to conserve document space.

**Table 14.3** Example Output File for the GWGRAB Program – Check Run Status

Current CFEST LPROG3 Run Status (Completed Time Steps)										
Analyte	1	2	3	4	5	6	7	8	9	10
Tc99	1163	1163	1163	1163	1163	1163	1163	1163	1163	1163
out of	1163	1163	1163	1163	1163	1163	1163	1163	1163	1163
H3	1163	1163	1163	1163	1163	1163	1163	1163	1163	1163
out of	1163	1163	1163	1163	1163	1163	1163	1163	1163	1163

#### 14.4.2 GWGRAB Example – Extract CFEST Results

GWGRAB can be used to extract mass balance information from CFEST runs. This task is accomplished by use of the EXTRACT BALANCE keyword in the GWGRAB keyword control file. Extracted data are placed into five output file types: influx, storage, outflux, decay, and error. The first four files help examine mass conservation in the runs; the change in storage equals influx minus outflux minus radioactive decay, whereas the error file reflects the accuracy of the calculation by displaying the error ratio (1.00 if mass is 100% conserved). All calculations are done at the end of each CFEST time step. An example of the GWGRAB keyword control file is provided in Table 14.4.

**Table 14.4** Example Keyword File for the GWGRAB Program – Extract CFEST Results

```

PATH "/home/ANALYSIS2/Initial3/gwgrab/" !Where to put extracted results
EXEDIR "/home/CODES/bin/" !Where the SAC executables are
EXTRACT CHECK !Report completed CFEST time steps.
ANALYTE "H3" !Report Tritium only.
REALIZATION 1 2 3 4 !Report realizations 1-4 only.
TIME ANNUAL !Mass balance reported at predefined times
OUTPUT RAW !Report raw data for each realization.
END

```

The extraction defined in Table 14.4 results in the following types of output files. Only the first few lines from each file are shown in Table 14.5.

**Table 14.5** Example Output File for the GWGRAB Program – Extract CFEST Results

#SAC Groundwater Analyte Influx Data, by Realization (kg)							
#SAC Data							
Extracted 9/10/02 12:28:16							
#SAC Problem : SAC Rev. 0 Initial3 Assessment							
Analyte							
Simulated Time (d)	Date & Time	ID	1	2	3	4	
1.83E+02	7/1/44 15:00	H3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
3.65E+02	12/31/44 6:00	H3	0.00E+00	0.00E+00	9.03E-19	0.00E+00	
5.48E+02	7/1/45 21:00	H3	1.38E-06	1.39E-06	1.11E-06	1.11E-06	

```
#SAC Groundwater Analyte Outflux Data, by Realization (kg)
#SAC Data
Extracted          9/10/02 12:28:16
#SAC Problem :    SAC Rev. 0 Initial3 Assessment
  Simulated Time      Analyte
    (d)           Date & Time      ID          1          2          3          4
  1.83E+02       7/1/44 15:00      H3          0.00E+00  0.00E+00  0.00E+00  0.00E+00
  3.65E+02       12/31/44 6:00     H3          0.00E+00  0.00E+00  9.42E-30  0.00E+00
  5.48E+02       7/1/45 21:00     H3         -4.51E-17 -1.45E-16 -4.40E-18  3.76E-18

#SAC Groundwater Analyte Decay Data, by Realization (kg)
#SAC Data
Extracted          9/10/02 12:28:16
                  SAC Rev. 0 Initial3
#SAC Problem :    Assessment
  Simulated Time      Analyte ID
    (d)           Date & Time      ID          1          2          3          4
  1.83E+02       7/1/44 15:00      H3          0.00E+00  0.00E+00  0.00E+00  0.00E+00
  3.65E+02       12/31/44 6:00     H3          0.00E+00  0.00E+00  0.00E+00  0.00E+00
  5.48E+02       7/1/45 21:00     H3          0.00E+00  0.00E+00  2.51E-20  0.00E+00
  7.31E+02       12/31/45 12:00    H3          3.90E-08  3.91E-08  3.12E-08  3.12E-08

#SAC Groundwater Analyte Balance Data, by Realization (kg)
#SAC Data
Extracted          9/10/02 12:28:16
#SAC Problem :    SAC Rev. 0 Initial3 Assessment
Simulated Time (d) Date & Time      Analyte ID          1          2          3          4
  1.83E+02       7/1/44 15:00      H3          0.00E+00  0.00E+00  0.00E+00  0.00E+00
  3.65E+02       12/31/44 6:00     H3          0.00E+00  0.00E+00  9.03E-19  0.00E+00
  5.48E+02       7/1/45 21:00     H3          1.38E-06  1.39E-06  1.11E-06  1.11E-06

#SAC Groundwater Analyte Mass Balance Error Fraction, by Realization
#SAC Data
Extracted          9/10/02 12:28:16
#SAC Problem :    SAC Rev. 0 Initial3 Assessment
Simulated Time (d) Date & Time      Analyte ID          1          2          3          4
  1.83E+02       7/1/44 15:00      H3          1.00E+00  1.00E+00  1.00E+00  1.00E+00
  3.65E+02       12/31/44 6:00     H3          1.00E+00  1.00E+00  1.00E+00  1.00E+00
  5.48E+02       7/1/45 21:00     H3          1.00E+00  1.00E+00  1.00E+00  1.00E+00
```

## **15.0 HIGHDOSE – Dose Extraction for ECEM**

### **15.1 Overview**

The HIGHDOSE code is designed to extract the largest impact calculated by the ECEM (see Eslinger et al. [2006b], Volume 2, Section 4.0) code over a suite of locations at each of several time steps. For example, it can determine the species with the highest dose along the riparian zone at a set of times.

#### **15.1.1 Location in the Processing Sequence**

The HIGHDOSE code must follow a run of the ECEM code.

#### **15.1.2 How the Code Is Invoked**

HIGHDOSE can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), HIGHDOSE executes in a DOS box. A run of HIGHDOSE is initiated by entering the following command line:

```
HIGHDOSE "Keyfilename"
```

Under the Linux operating system HIGHDOSE is executed through any of the following Bourne Shell or C Shell commands:

```
highdose-1.exe "Keyfilename"
```

For these commands, “HIGHDOSE.EXE” or “highdose.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHDOSE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHDOSE cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **15.1.3 Memory Requirements**

The HIGHDOSE program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

### **15.2 File Definitions**

The HIGHDOSE program reads three input files and writes to two output files. These files are described in the following paragraphs.

#### **15.2.1 Input Files**

The input files for the HIGHDOSE program are a control keyword file, a species definition file, and a details result file output by the ECEM code.

### 15.2.1.1 HIGHDOSE Keyword File

The HIGHDOSE keyword file contains control information. An example file is provided in Table 15.1. Detailed definitions of the keywords are provided in Section 15.3.

**Table 15.1** Example Keyword File for HIGHDOSE

```
! Keyword file for the FilleCDA program
REPORT "Test.Rpt"
TITLE "Testing of the HighDose Code Using a ECEM Rev. 1 run"
USER "Paul W. Eslinger"
FILE SPECIES "Species.csv"
FILE ECEM "Ecem_Det.Csv"
FILE RESULTS "Test.Csv"
REALIZATION 2
MEMORY TABLEROW=3 REALIZATION=25 RECORDS=1000
!VERBOSE
SOLUTION "SUMRAD"
ANALYTE "-Rads-"
SOIL "SOSW"
! The following keywords allow multiple choices
LOCATION "TH0001" "TH0003" "TH0015" "TH0017"
        "TH0002" "TH0005" "TH0006"
TIME 1950 1955 1960 1965
END
```

### 15.2.1.2 Species Definition File

The HIGHDOSE program requires a species definition file. This file contains the species type, species ID, and species name for every species for which data are provided in the ECEM (see Eslinger et al. [2006b], Volume 2, Section 4.0) results file. This information is required, but is used mainly for labeling purposes. This file is a text file written using a comma-separated format. An example file is provided in Table 15.2.

**Table 15.2** Example Species Definition File for HIGHDOSE

```
"TA","AMCOOT","American coot"
"TA","AMCOUP","American coot - upland"
"TA","EGGS","Chicken eggs - upland"
"TA","BEAVER","Beaver"
"QA","CARP","Carp"
"TA","COYOTE","Coyote"
"QA","DAPMAG","Daphnia magna"
"TP","DENS DG","Dense sedge"
"TP","DENSUP","Dense sedge - upland"
"TP","FUNGI","Fungi"
"TP","MULBRY","Mulberry"
"TP","MULBUP","Mulberry - upland"
"TA","MULDER","Mule deer"
"TP","GRAIN","Generic grain species"
"TP","POTATO","Potato"
"TA","MILKCW","Milk cow"
"QP","PERPHY","Periphyton"
"QP","PHYPLK","Phytoplankton"
```

### 15.2.1.3 ECEM Results File

The third input file for the HIGHDOSE program is nominally generated with a run of the ECEM code. This file contains data output from ECEM under the control of the DETAILS keyword in ECEM.

### 15.2.2 Output Files

The HIGHDOSE program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the extracted information. The first few lines from an example results file is provided in Table 15.3.

**Table 15.3** Example Results File from the HIGHDOSE Code

```

"Code Name:", "HighDose"
"Code Version:", "2.0.A.00"
"Code Date:", " 4 Mar 2003"
"Run ID:", "20030904132249"
"Run Title:", "Testing of the HighDose Code Using a ECEM Rev. 1 run"
"User Name:", "Paul W. Eslinger"

"Realization 1 is requested."

"Results for location ID = TH0011 year = 1955 solution = SUMRAD soil type = SOSW and
analyte type = -Rads-"
"Terrestrial Plants", " ", "Terrestrial Animals", " ", "Aquatic Plants", " ", "Aquatic
Animals"
"Specie", "Dose", "Specie", "Dose", "Specie", "Dose", "Specie", "Dose"
"MULBUP", 8.090E+04, "AMCOUP", 5.339E+04, " ", 0.000E+00, " ", 0.000E+00
"DENSUP", 1.950E+04, "MILKCW", 4.703E+04, " ", 0.000E+00, " ", 0.000E+00
"POTATO", 1.950E+04, "MULDER", 2.893E+04, " ", 0.000E+00, " ", 0.000E+00

"Results for location ID = TH0014 year = 1955 solution = SUMRAD soil type = SOSW and
analyte type = -Rads-"
"Terrestrial Plants", " ", "Terrestrial Animals", " ", "Aquatic Plants", " ", "Aquatic
Animals"
"Specie", "Dose", "Specie", "Dose", "Specie", "Dose", "Specie", "Dose"
"MULBUP", 3.318E+04, "AMCOUP", 9.489E+04, " ", 0.000E+00, " ", 0.000E+00
"POTATO", 8.181E+03, "COYOTE", 9.202E+04, " ", 0.000E+00, " ", 0.000E+00
"DENSUP", 8.181E+03, "MILKCW", 7.955E+04, " ", 0.000E+00, " ", 0.000E+00

```

The most typical method for looking at the results of the HIGHDOSE code is to reformat the comma-separated results into a table. An example of such a result is provided in Table 15.4.

**Table 15.4** Reformatted Output From the HIGHDOSE Code

Results for location ID = NAPLSH year = 2500 solution = SUMRAD and analyte type = -Rads-							
Terrestrial Plants		Terrestrial Animals		Aquatic Plants		Aquatic Animals	
Specie	Dose	Specie	Dose	Specie	Dose	Specie	Dose
TULE	9.75E-11	CLFSWL	1.75E-08	PERPHY	7.72E-09	MAYFLY	3.57E-08
BLCTWD	8.81E-11	AMCOOT	7.94E-09	WMLFOL	5.74E-09	CRYFSH	1.17E-08
MULBRY	8.81E-11	BUFLHD	5.32E-09	PHYPLK	6.43E-12	CPBLSN	7.99E-09
CYLCRS	4.67E-11	COMSNP	5.01E-09		0.00E+00	SALMJV	5.53E-09
RCANGS	3.36E-11	RACoon	3.20E-09		0.00E+00	RTRTJV	5.53E-09

## 15.3 Keyword Definitions for HIGHDOSE

In general, the keywords for the HIGHDOSE code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 15.3.1 ANALYTE Keyword for HIGHDOSE

The ANALYTE keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ "quote 1" ]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string "-Rads-" if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANALYTE "C14"  
ANALYTE "-Rads-"
```

### 15.3.2 END Keyword for HIGHDOSE

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 15.3.3 FILE Keyword for HIGHDOSE

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"} {modifier3="quote3"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the ECEM details file is associated with the modifier ECEM. The name of the input file defining ecological species is associated with the modifier SPECIES. The name of the output file from HIGHDOSE is associated with the modifier RESULTS. Example file keywords that define these three files are the following:

```
FILE SPECIES "Species.csv"  
FILE ECEM "ECEM_SWEIS_det_Bg.Csv"  
FILE RESULTS "High_Rads_Bg.Csv"
```

### 15.3.4 LOCATION Keyword for HIGHDOSE

The LOCATION keyword is used to define the set of locations for which data will be processed. The following is this keyword's syntax:

```
LOCATION ["quote 1"] {"quote2" ... "quoten"}
```

One or more quote strings must contain valid location IDs from locations specified in the ESD keyword file. Data for a given time step is scanned over the entire set of specified locations. An example LOCATION keyword defining ten locations is the following:

```
LOCATION
"UH0001" "UH0002" "UH0003" "UH0004" "UH0005"
"UH0006" "UH0007" "UH0008" "UH0009" "UH0010"
```

### 15.3.5 MEMORY Keyword for HIGHDOSE

The MEMORY keyword is used to define some variables that determine the amount of memory used in the code. The following is this keyword's syntax:

```
MEMORY [TABLEROW=N1] [REALIZAT=N2] [RECORDS=N3]
```

The modifiers associated with the MEMORY keyword are described in Table 15.5.

**Table 15.5** Modifiers Associated with the MEMORY Keyword in HIGHDOSE

Modifier	Description
TABLEROW	The numerical value associated with the TABLEROW modifier defines the number of rows to use in the output table. A value of 3 to 5 is suggested.
REALIZAT	The numerical value associated with the REALIZAT modifier defines the number of realizations used in the run of ECEM that produced the detailed data file being analyzed.
RECORDS	The numerical value associated with the RECORDS modifier defines the number of records that may be matched in the detailed data file for a single year and location combination. Typically this value can be set to the number of species.

An example keyword where the user specifies five rows for the output table for a deterministic run of ECEM that utilized 100 species is the following:

```
MEMORY TABLEROW=5 REALIZATION=1 RECORDS=100
```

### 15.3.6 REALIZAT Keyword for HIGHDOSE

The REALIZAT (or REALIZATION) keyword defines the single realization for which data are to be extracted. The following is this keyword's syntax:

```
REALIZATION value1
```

The integer value1 has a minimum value of 1 and a maximum of the number of realizations performed by the ECEM code. The following keyword record extracts data for realization number 10:

REALIZAT 10

### 15.3.7 REPORT Keyword for HIGHDOSE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 15.3.8 SOIL Keyword for HIGHDOSE

The SOIL keyword is used to define the solution type for which data will be processed. The following is this keyword's syntax:

```
SOIL [ "quote 1" ]
```

The single quote string is case sensitive and six characters in length. The modifiers associated with the SOILTYPE keyword are given in Table 15.6.

**Table 15.6** Modifiers Associated with the SOIL Keyword in HIGHDOSE

Modifier	Description
NONE	Applies to all aquatic locations
SORP	Riparian locations
SODR	Dry (non irrigated) upland locations
SOGW	Groundwater irrigated upland locations
SOSW	Surface water irrigated upland locations

An example of this keyword is the following:

```
SOIL "SOGW"
```

### 15.3.9 SOLUTION Keyword for HIGHDOSE

The SOLUTION keyword is used to define the solution type for which data will be processed. The following is this keyword's syntax:

```
SOLUTION [ "quote 1" ]
```

The single quote string is case sensitive and six characters in length. The modifiers associated with the SOIL keyword are given in Table 15.7.



**Table 15.7** Modifiers Associated with the SOLUTION Keyword in HIGHDOSE

Modifier	Description
BURDEN	Body burdens (pCi or µg/kg wet wt)
DOSRAD	Radioactive dose (rem)
BMTISS	Ratios to tissue benchmarks (unitless)
SUMRAD	Sum of radioactive dose (rem)
CONCEN	Media concentrations (see MEDIA keyword modifiers for units)

An example of this keyword is the following:

```
SOLUTION "BURDEN"
```

### 15.3.10 TIME Keyword for HIGHDOSE

The TIME keyword identifies the times at which the calculations are to be performed. The following is this keyword's syntax:

```
TIME [T1] {T2} ... {Tn}
```

The numerical entries T1, T2, ..., Tn are the times (whole number years) when outputs are desired. These times must be a subset of the times at which environmental data were computed and stored by the inventory, release, and transport modules. Only one TIME keyword should be entered. The following is an example TIME keyword that requests output for the three years 2020, 2075, and 3014:

```
TIME 2020 2075 3014
```

### 15.3.11 TITLE Keyword for HIGHDOSE

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the HIGHDOSE code."
```

### 15.3.12 USER Keyword for HIGHDOSE

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### **15.3.13 VERBOSE Keyword for HIGHDOSE**

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

## **16.0 HIGHIMPACT – Maximum Impact Extraction for HUMAN**

### **16.1 Overview**

The HIGHIMPACT code is designed to extract the largest impact result calculated by the HUMAN code over a suite of locations at each of several time steps. For example, it can provide the largest dose to an individual outside the Hanford core zone as a function of time.

#### **16.1.1 Location in the Processing Sequence**

The HIGHIMPACT code must follow a run of the HUMAN code.

#### **16.1.2 How the Code Is Invoked**

HIGHIMPACT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), HIGHIMPACT executes in a DOS box. A run of HIGHIMPACT is initiated by entering the following command line:

```
HIGHIMPACT "Keyfilename"
```

Under the Linux operating system HIGHIMPACT is executed through any of the following Bourne Shell or C Shell commands:

```
highimpact-1.exe "Keyfilename"
```

For these commands, “HIGHIMPACT .EXE” or “highimpact-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHIMPACT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHIMPACT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **16.1.3 Memory Requirements**

The HIGHIMPACT program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

### **16.2 File Definitions**

The HIGHIMPACT program reads two input files and writes two output files. These files are described in the following sections.

#### **16.2.1 Input Files**

The HIGHIMPACT program reads a control keyword file and an impact details file written by the HUMAN code. The HIGHIMPACT keyword file contains control information. An example file is provided in Table 16.1. Detailed definitions of the keywords are provided in Section 16.3.

**Table 16.1** Example Keyword File for the HIGHIMPACT Program

```
! Keyword file for HIGHIMPACT
REPORT "Test.Rpt"
TITLE "Testing of the HighImpact Code Using a HUMAN Rev. 1 run"
USER "Paul W. Eslinger"
FILE HUMAN "Farmer_CAI_median_Dtl.csv"
FILE RESULTS "Test.Csv"
REALIZAT HUMAN=1 SINGLE=1
! Locations can be multiply defined
LOC_ID "UH1082" "UH1083"
! Times can be multiply defined
TIME 1990 1992
! The following solutions must be uniquely defined
S_TYPE "GWAT"
ANA_ID "C14"
R_TYPE "CON"
VERBOSE
END
```

## 16.2.2 Output Files

The HIGHIMPACT program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the information about the highest impact over a set of locations for every year. An example results file is provided in Table 16.2.

**Table 16.2** Example Results File from the HIGHIMPACT Code

```
"Code Name:", "HighImpact"
"Code Version:", "1.00.005"
"Code Date:", "22 Mar 2006"
"Run ID:", "20060515152050"
"Run Title:", "Maximum dose for the upland Residential Farmer scenario"
"User Name:", "Carmen Arimescu"
"Solution type:", "SINGLE = 1", "COMBIN", "RAD", "SUMDOSE"
1945, "UH2709", 8.29183E-03
1950, "UH2709", 1.85205E-02
1955, "UH0858", 1.50036E+00
1960, "UH1642", 9.24540E-01
1965, "UH1643", 3.96217E+00
1970, "UH1643", 4.82087E+00
1975, "UH1644", 3.36062E+00
1980, "UH1644", 1.90950E+00
1985, "UH1643", 1.12953E+00
1990, "UH1642", 6.65234E-01
1991, "UH1642", 5.95870E-01
1992, "UH1642", 5.27415E-01
1993, "UH1642", 4.52947E-01
1994, "UH1643", 3.87738E-01
1995, "UH1643", 3.63506E-01
1996, "UH1643", 3.40908E-01
1997, "UH1643", 3.18804E-01
1998, "UH1709", 2.94928E-01
1999, "UH1709", 2.77404E-01
2000, "UH1709", 2.60285E-01
```

## 16.3 Keyword Definitions for the HIGHIMPACT Code

In general, the keywords for the HIGHIMPACT code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

The keyword selections require the user to specify a set of possible values for the first five data values on each line in the details file written by the HUMAN code. The relative position of the values in the details file and the associated keyword is the following:

1. TIME – Multiple time definitions are allowed on a single TIME keyword.
2. LOCATION – Multiple location definitions are allowed on a single LOCATION keyword.
3. ANA\_ID – A single solution is required for this keyword.
4. R\_TYPE – A single solution is required for this keyword.
5. S\_TYPE – A single solution is required for this keyword.

Values for the data in positions 3, 4, and 5 depend on the specified solutions. An easy way to choose the correct combination is to examine the results data file that is to be processed.

### 16.3.1 ANA\_ID Keyword for HIGHIMPACT

The ANA\_ID keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword's syntax

```
ANA_ID [ "quote 1" ]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string "-Rads-" if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANA_ID "C14"  
ANA_ID "-Rads-"
```

### 16.3.2 END Keyword for HIGHIMPACT

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 16.3.3 FILE Keyword for HIGHIMPACT

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the human details file is associated with the modifier HUMAN. The name of the output file from HIGHIMPACT is associated with the modifier RESULTS. Example file keywords that define these two files are the following:

```
FILE HUMAN "Farmer_CA1_median_Dtl.csv"
FILE RESULTS "Farmer_High.csv"
```

### 16.3.4 LOC\_ID Keyword for HIGHIMPACT

The LOC\_ID keyword is used to define the set of locations for which data will be processed. The following is this keyword's syntax:

```
LOC_ID ["quote 1"] {"quote2" ... "quoten"}
```

One or more quote strings must contain valid location IDs from locations specified in the ESD keyword file. Data for a given time step is scanned over the entire set of specified locations. An example LOC\_ID keyword defining ten locations is the following:

```
LOC_ID
"UH0001" "UH0002" "UH0003" "UH0004" "UH0005"
"UH0006" "UH0007" "UH0008" "UH0009" "UH0010"
```

### 16.3.5 REALIZAT Keyword for HIGHIMPACT

The REALIZAT (or REALIZATION) keyword is used to define the realizations for which data will be processed. The following is this keyword's syntax:

```
REALIZAT [HUMAN=N1] [MAXIMUM|MEAN|SINGLE=N2]
```

The modifiers associated with the REALIZAT keyword are described in Table 16.3. Only one of the MAXIMUM, MEAN, or SINGLE modifiers are allowed during a run of the code.

**Table 16.3** Modifiers Associated with the REALIZAT Keyword in HIGHIMPACT

Modifier	Description
HUMAN	The numerical value associated with the HUMAN modifier defines the number of realizations contained in the details file written by the HUMAN code.
MAXIMUM	This modifier indicates that the maximum over all realizations will be used in determining the highest impact at each location.
MEAN	This modifier indicates that the arithmetic mean over all realizations will be used in determining the highest impact at each location.
SINGLE	This modifier indicates that data from a single realization will be used in determining the highest impact at each location. The realization number to use is provided in the numerical value associated with the SINGLE modifier.

An example keyword where the user specifies using the arithmetic mean for a data file containing 100 realizations of data is the following:

```
REALIZAT HUMAN=100 MEAN
```

An example keyword where the user specifies using the data for realization 23 from a data file containing 100 realizations of data is the following:

```
REALIZAT HUMAN=100 SINGLE=23
```

### 16.3.6 REPORT Keyword for HIGHIMPACT

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 16.3.7 R\_TYPE Keyword for HIGHIMPACT

The R\_TYPE keyword is used to partially define the solution type for which data will be processed. The following is this keyword's syntax:

```
R_TYPE [ "quote 1" ]
```

The single quote string is case sensitive and six characters in length. The modifiers associated with the analyte type for the R\_TYPE keyword are given in Table 16.4.

**Table 16.4** Modifiers Associated with the R\_TYPE Keyword in HIGHIMPACT

Modifier	Description
ALL	The analyte field is used for a quantity that is a sum over analytes.
CAR	The analyte is a carcinogenic chemical.
CONMEDIA	The analyte field is used to select an environmental medium.
CONFOODS	The analyte field is used to select a food species.
HAZ	The analyte is a hazardous chemical.
RAD	The analyte is a radionuclide.

An example of this keyword is the following:

```
R_TYPE "RAD"
```

### 16.3.8 S\_TYPE Keyword for HIGHIMPACT

The S\_TYPE keyword is also used to define the solution type for which data will be processed. The following is this keyword's syntax:

```
S_TYPE [ "quote 1" ]
```

The single quote string is case sensitive and is two to seven characters in length. The modifiers associated with the S\_TYPE keyword are given in Table 16.5.

**Table 16.5** Modifiers Associated with the S\_TYPE Keyword in HIGHIMPACT

Modifier	Description
<b>S_TYPE Modifiers Associated with R_TYPE CONMEDIA</b>	
AIRC	Air concentrations
GWAT	Groundwater concentrations
SEDI	Sediment concentrations
SEEP	Seep water concentrations
SODR	Soil concentrations, no irrigation
SOGW	Soil concentrations, groundwater irrigated
SORP	Riparian soil concentrations
SOSW	Soil concentrations, surface water irrigated
SWAT	Surface water concentrations
<b>S_TYPE Modifiers Associated with R_TYPE CONFOODS</b>	
BIRD	Food concentration, birds
EGGS	Food concentration, eggs
FISH	Food concentration, fish
FISH_2	Food concentration, 2 <sup>nd</sup> fish type
FISH_3	Food concentration, 3 <sup>rd</sup> fish type
FRUIT	Food concentration, fruit
GRAIN	Food concentration, grain
LEAFVEG	Food concentration, leafy vegetables
MEAT	Food concentration, meat
MILK	Food concentration, milk
ROOTVEG	Food concentration, root vegetables
<b>S_TYPE Modifiers Associated with R_TYPE CAR, HAZ, RAD</b>	
ANADOSE	Analyte dose
ANARISK	Analyte risk (CAR and RAD)
ANAHQ	Analyte hazard quotient (HAZ)
DOSEING	Ingestion dose (CAR, HAZ and RAD)
DOSEINH	Inhalation dose (CAR, HAZ and RAD)
DOSEEXT	External dose (RAD)
DOSEDER	Dermal dose (CAR and HAZ)
HQING	Ingestion hazard quotient (HAZ)
HQINH	Inhalation hazard quotient (HAZ)
HQDER	Dermal hazard quotient (HAZ)
POPDOSE	Population dose (RAD)
POPRISK	Population risk (RAD)
RISKING	Ingestion risk (CAR and RAD)
RISKINH	Inhalation risk (CAR and RAD)
RISKEXT	External risk (RAD)
RISKDER	Dermal risk (CAR)
SUMDOSE	Dose summed over analytes (CAR and HAZ)
SUMHQ	Hazard quotient summed over analytes (HAZ)
<b>S_TYPE Modifiers Associated with R_TYPE ALL</b>	
SUMRISK	Risk summed over carcinogenic chemicals and radionuclides



An example of this keyword is the following:

```
S_TYPE "ANADOSE"
```

### 16.3.9 TIME Keyword for HIGHIMPACT

The TIME keyword identifies the times at which the calculations are to be performed. The following is this keyword's syntax:

```
TIME [T1] {T2} ... {Tn}
```

The numerical entries T1, T2, ..., Tn are the times (whole number years) when outputs are desired. These times must be a subset of the times at which environmental data were computed and stored by the inventory, release, and transport modules. Only one TIME keyword should be entered. The following is an example TIME keyword that requests output for the three years 2020, 2075, and 3014:

```
TIME 2020 2075 3014
```

### 16.3.10 TITLE Keyword for HIGHIMPACT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the HIGHIMPACT code."
```

### 16.3.11 USER Keyword for HIGHIMPACT

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 16.3.12 VERBOSE Keyword for HIGHIMPACT

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```



## 17.0 HIGHMEDIA – Extraction of Maximum Media Concentrations

### 17.1 Overview

The HIGHMEDIA code is designed to extract the maximum media concentrations over a set of specified locations and times for data contained in an ECDA file. For example, it can provide the highest groundwater concentration for a single analyte outside the Hanford core zone as a function of time.

#### 17.1.1 Location in the Processing Sequence

The HIGHMEDIA code reads data from an ECDA file. Thus, the transport codes must have been executed and the media entered in the ECDA files before HIGHMEDIA can be used.

#### 17.1.2 How the Code Is Invoked

HIGHMEDIA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), HIGHMEDIA executes in a DOS box. A run of HIGHMEDIA is initiated by entering the following command line:

```
HIGHMEDIA "Keyfilename"
```

Under the Linux operating system HIGHMEDIA is executed through any of the following Bourne Shell or C Shell commands:

```
highmedia-1.exe "Keyfilename"
```

For these commands, “HIGHMEDIA.EXE” or “highmedia-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHMEDIA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHMEDIA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### 17.1.3 Memory Requirements

The HIGHMEDIA program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

### 17.2 File Definitions

The HIGHMEDIA program reads two input files and writes two output files. These files are described in the following sections.

## 17.2.1 Input Files

The HIGHMEDIA program reads a control keyword file and an ECDA file. The HIGHMEDIA keyword file contains control information. An example file is provided in Table 17.1. Detailed definitions of the keywords are provided in Section 17.3.

**Table 17.1** Example Keyword File for the HIGHMEDIA Program

```
REPORT "CPO_Alternate_High_GWAT.Rpt"
USER "Paul W. Eslinger"
FILE ESD "\SAC\ANALYSES\CPO_Alternate\ESD_CPO_Alternate.key"
!VERBOSE
ENDINIT

CASE
TITLE "Maximum Concentrations Tc99 at Core Zone Boundary"
FILE RESULTS "CPO_Alternate_GWAT_Tc99_CORE.csv"
ANALYTE "Tc99"
REALIZAT SINGLE=1
MEDIA "GWAT"
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
TIMES ALL
LOCATION LIST ! Boundary locations for Core Zone
"UH0722" "UH0724" "UH0725" "UH0736" "UH0739" "UH0748" "UH0749"
"UH0752" "UH0754" "UH0755" "UH0756" "UH0757" "UH0758" "UH0759"
"UH0762" "UH0764" "UH0765" "UH0766" "UH0767" "UH0768" "UH0769"
ENDCASE

CASE
TITLE "Maximum Concentrations Tc99 at River Shore Boundary"
FILE RESULTS "CA1_median_GWAT_Tc99_CORE_RIVER.csv"
ANALYTE "Tc99"
REALIZAT SINGLE=1
MEDIA "GWAT"
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
TIMES ALL
LOCATION LIST ! Boundary locations for the river shore (Hanford Side)
"RHP001","RHP003","RHP004","RHP005","RHP006","RHP007","RHP008"
"RHP011","RHP013","RHP014","RHP015","RHP016","RHP017","RHP018"
"RHP021","RHP023","RHP024","RHP025","RHP026","RHP027","RHP028"
ENDCASE

END
```

## 17.2.2 Output Files

The HIGHMEDIA program writes two or more output files. One file is a text report file that contains information about the run of the code, including any error messages. The other output file (or files) is the results file containing the information about the highest result over a set of locations for every year. A separate output file is written for every case definition. An example results file is provided in Table 17.2.

**Table 17.2** Example Results File from the HIGHMEDIA Code

```
"Code Name:", "HighMedia"
"Code Version:", "2.00.A.1"
"Code Date:", "16 Aug 2004"
"Run ID:", "20040817110129"
"Run Title:", "HighMedia Using a ECDA Files Rev. 1 run"
"User Name:", "Paul W. Eslinger"
"File Name:", "H:\CA1_median\ecda_save\Tc99_CA1_median.dat"
"Solution type:", "MEDIAN"
"Analyte ID:", "Tc99"
"Media ID:", "GWAT"
"Year", "Solution", "Location ID", "Easting", "Northing"
1990, 4.52490E-07, "UH1230", 5.74522E+05, 1.35992E+05
1991, 3.61323E-07, "UH1230", 5.74522E+05, 1.35992E+05
1992, 2.89056E-07, "UH1230", 5.74522E+05, 1.35992E+05
```

## 17.3 Keyword Definitions for HIGHMEDIA

The keywords for the HIGHMEDIA code are grouped into an initial section and one or more case groups. The initial section must begin with the REPORT keyword and must end with an ENDINIT keyword. Each analysis case must start with a CASE keyword and end with an ENDCASE keyword. The END keyword must be the last keyword in the file.

### 17.3.1 Keywords in the Initial Section for HIGHMEDIA

The initial section of keywords starts with the REPORT keyword and ends with the ENDINIT keyword. The run will error terminate if these conditions are not met.

#### 17.3.1.1 ENDINIT Keyword for HIGHMEDIA

The ENDINIT keyword signifies the end of the initial section of the keyword data. The following is this keyword's syntax:

```
ENDINIT
```

#### 17.3.1.2 FILE Keyword for HIGHMEDIA

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax in the initial keyword section:

```
FILE [ESD="quote1"]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The name of the ESD control keyword file is associated with the modifier ESD. An example keyword is the following:

```
FILE ESD "C:\SAC\SAC_1\Analysis_2004\CA1_Median\ESD_CA1_median.key"
```

### 17.3.1.3 REPORT Keyword for HIGHMEDIA

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. An example keyword is the following:

```
REPORT "/SAC/C/Test.rpt"
```

### 17.3.1.4 USER Keyword for HIGHMEDIA

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 17.3.1.5 VERBOSE Keyword for HIGHMEDIA

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

## 17.3.2 Keywords for Specific Cases for HIGHMEDIA

The definition of an extraction case starts with the CASE keyword and ends with the ENDCASE keyword. Multiple extraction cases can be defined.

### 17.3.2.1 ANALYTE Keyword for HIGHMEDIA

The ANALYTE keyword is used to define the analyte for which data will be processed for a single case. The following is this keyword's syntax:

```
ANALYTE [ "quote 1" ]
```

The single quote string must be a single analyte ID from the set of the analytes in the ESD keyword file. An example of this keyword using the analyte technetium-99 is the following:

```
ANALYTE "Tc99"
```

### 17.3.2.2 CASE Keyword for HIGHMEDIA

The CASE keyword signifies the beginning of the definition of a specific case of keyword data. The following is this keyword's syntax:

```
CASE
```

### 17.3.2.3 ENDCASE Keyword for HIGHMEDIA

The ENDCASE keyword signifies the end of the definition of a specific case of keyword data. The following is this keyword's syntax:

```
ENDCASE
```

### 17.3.2.4 FILE Keyword for HIGHMEDIA

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax in a case definition:

```
FILE [RESULTS="quote1"]
```

File names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. Exactly one FILE keyword is required for every case in the keyword file. The name of the output file from HIGHMEDIA is associated with the modifier RESULTS. An example keyword that defines a results file is the following:

```
FILE RESULTS "Test.Csv"
```

### 17.3.2.5 LOCATION Keyword for HIGHMEDIA

The LOCATION keyword is used to define the set of locations for which data will be processed for a single case. The following is this keyword's syntax:

```
LOCATION [ALL| LIST ["quote 1"] {"quote2" ... "quoten"}]
```

All locations in the ESD keyword file are used when the modifier ALL is present. A list of specific sites can be defined by entering the modifier LIST and a set of quote strings identifying the site IDs of interest. The specific locations entered must be defined in the ESD keyword file. All locations are included in the first example shown below. The second example defines a list of six specific sites.

```
LOCATION ALL  
LOCATION LIST "UH1000" "UH1001" "UH1003" "UH1004" "UH1005" "UH1230"
```

### 17.3.2.6 MEDIA Keyword for HIGHMEDIA

The MEDIA keyword is used to define the media for which data will be processed for a single case. The following is this keyword's syntax:

```
MEDIA ["quote 1"]
```

The single quote string is case sensitive and four characters in length. The modifiers associated with the FILE keyword are given in Table 17.3.

**Table 17.3** Modifiers Associated with the MEDIA Keyword in HIGHMEDIA

Modifier	Description
GWAT	concentrations in groundwater ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEEP	concentrations in seep water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SWAT	concentrations in surface water (river) ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
PWAT	concentrations in river bottom pore water ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
SEDI	concentrations in river bottom sediment ( $\text{Ci}/\text{kg}_{\text{sediment}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{sediment}}$ )
SORP	concentrations in riparian zone soil (land surface) ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SODR	concentrations in upland soil (land surface) with no irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOGW	concentrations in upland soil (land surface) with groundwater irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
SOSW	concentrations in upland soil (land surface) with surface water irrigation ( $\text{Ci}/\text{kg}_{\text{soil}}$ or $\text{kg}_{\text{analyte}}/\text{kg}_{\text{soil}}$ )
AIRC	concentrations in air ( $\text{Ci}/\text{m}^3$ or $\text{kg}/\text{m}^3$ )
AIRD	air deposition rates ( $\text{Ci}/\text{m}^2/\text{yr}$ or $\text{kg}/\text{m}^2/\text{yr}$ )

An example of this keyword that uses concentrations in upland soil with surface water irrigation is the following:

MEDIA "SOSW"

### 17.3.2.7 REALIZATION Keyword for HIGHMEDIA

The REALIZAT keyword is used to define the realizations for which data will be processed and the summary technique to be used for a single case. The following is this keyword's syntax:

REALIZAT [MAXIMUM | MEAN | MEDIAN | SINGLE=N1]

The modifiers associated with the REALIZAT keyword are described in Table 17.4. Only one of the MAXIMUM, MEAN, MEDIAN or SINGLE modifiers are allowed during a run of the code.

**Table 17.4** Modifiers Associated with the REALIZAT Keyword in HIGHMEDIA

Modifier	Description
MAXIMUM	This modifier indicates that the maximum over all realizations will be used in determining the highest impact at each location.
MEDIAN	This modifier indicates that the median over all realizations will be used in determining the highest impact at each location.
MEAN	This modifier indicates that the arithmetic mean over all realizations will be used in determining the highest result at each location.
SINGLE	This modifier indicates that data from a single realization will be used in determining the highest result at each location. The realization number to use is provided in the numerical value associated with the SINGLE modifier.



Example keywords where the user specifies using the single realization number 5, the arithmetic mean of all realizations, and the median of all realizations are the following:

```
REALIZAT SINGLE=5
REALIZAT MEAN
REALIZAT MEDIAN
```

### 17.3.2.8 TIMES Keyword for HIGHMEDIA

The TIMES keyword identifies the times (years) at which the calculations are to be performed for a single case. The following is this keyword's syntax:

```
TIMES [ALL | LIST [T1] {T2} ... {Tn}]
```

All times in the ESD keyword file are used when the modifier ALL is present. A list of specific times can be defined by entering the modifier LIST and a set of numerical values identifying the years of interest. The specific years entered must be defined in the ESD keyword file. All times are included in the first example shown below. The second example defines a list of six years.

```
TIMES ALL
TIMES LIST 2020 2075 3014 3050 4000 12050
```

### 17.3.2.9 TITLE Keyword for HIGHMEDIA

The TITLE keyword is used to define a single-line problem title for a single case. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the HIGHMEDIA code."
```

### 17.3.2.10 UNITS Keyword for HIGHMEDIA

The UNITS keyword is used to define the data units for the analyte to be processed. The following is this keyword's syntax:

```
UNITS [OUTPUT="quote 1"] [FACTOR=N1]
```

The media data in the ECDA files have a predefined set of units (see the MEDIA keyword in Section 17.3.2.6). This keyword allows the user to change the units upon output. The single quote string associated with the modifier OUTPUT contains the units label for the output data (limit of 10 characters). The numerical value associated with the modifier FACTOR is the multiplicative factor required to change from input units to output units. Use a value of 1 if the input units are to be preserved. Example keywords that use groundwater concentrations of technetium-99 and convert the units from Ci/m<sup>3</sup> to pCi/L upon output are the following:

```
ANALYTE "Tc99"  
MEDIA "GWAT"  
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
```

### 17.3.3 Concluding (END) Keyword for HIGHMEDIA

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

## 18.0 HTWOS\_TDP – Tank Inventory Data Processor

### 18.1 Overview

This processor analyzes data files provided by the Hanford Tank Waste Operations Simulator (HTWOS) project describing the processing of waste in the 177 tanks at Hanford. The data are provided in a series of files, one file for each year from about 2003 through the end of processing (about 2030). The files give the total amount present at the end of each year, without any decay correction. The differences between files for two successive years indicate changes occurring during that year.

This program reads the HTWOS data files, determines actions taken during each year, and generates disposal streams from processing of tank waste. Output from the program is a data file ready for importing into the inventory database. Because the HTWOS files do not include radioactive decay corrections, the HTWOS\_TDP program incorporates decay calculations from the decay date for the input files to the end of the year of disposal.

The HTWOS files include the quantity in each tank and the quantity in each product and waste disposal stream (no waste stream generation is necessary). The product and waste disposal streams in the current HTWOS input files are listed in Table 18.1, along with the assumed waste type and disposal site acronym. The table also indicates the source of the data in the HTWOS files to be used to generate each waste stream.

**Table 18.1** Waste Streams Generated from Processing of Tank Wastes

Waste Stream	Waste Type	Disposal Site Acronym
HTWOS-SPENT-RESIN	cement	H2-SPENT-RESIN
LAW-HEPA1	cement	H2-LAW-HEPA1
LAW-HEPA2	cement	H2-LAW-HEPA2
LAW-VOC-SCRUB	cement	H2-LAW-VOC-SCRB
HLW-HEPA1	cement	H2-HLW-HEPA1
HLW-HEPA2	cement	H2-HLW-HEPA2
HLW-VOC-SCRUB	cement	H2-HLW-VOC-SCRB
HLW-AG-MORDENITE-COL	cement	H2-AG-M-COL
CONTACT-SUPP-TRU-PACKAGES	store	H2-CHTRU-PKGS
REMOTE-SUPP-TRU-PACKAGES	store	H2-RHTRU-PKGS
ETF-SOLID-EFFLUENT	cement	H2-ETF-SOLID
ETF-LIQUID-EFFLUENT	liquid	H2-ETF-LIQUID
BV-PRODUCT	glass	H2-BV-PRODUCT
HTWOS-HLW-CANISTERS	glass	H2-HLW-CANISTER
FAILED-HLW-MELTERS	glass	H2-HLW-MELTERS
FAILED-LAW-MELTERS	glass	H2-LAW-MELTERS
HTWOS-LAW-CANISTERS	glass	H2-LAW-CANISTER
BV-STACK	gas	H2-BV-STACK
WTP-STACK	gas	H2-WTP-STACK
HTWOS-LAW-OFFGAS-STACK	gas	H2-LAW-STACK
SUPP-TRU-STACK	gas	H2-TRU-STACK
242-A-STACK	gas	H2-242A-STACK
Tank residual waste after tank closure	res	241-“Tank ID”

The data in the HTWOS annual files gives the cumulative amount of each contaminant present at the end of the year. For each pair of years, the difference between annual values is the change in amounts, representing the amount of contaminant that goes from one source to one of the product streams or waste disposal streams. When the tank amounts decrease, there should be a corresponding increase in one or more of the product and/or waste disposal streams.

### 18.1.1 Location in the Processing Sequence

The HTWOS\_TDP code provides data files to be loaded into the inventory database. It must be run before the inventory code can be run.

### 18.1.2 How the Code Is Invoked

HTWOS\_TDP runs in a DOS box under the Windows operating system. A run of HTWOS\_TDP is initiated by entering the following command line:

```
HTWOS_TDP
```

The HTWOS\_TDP utility code does not require any additional command line inputs. The assumption is that all files used by the code will reside in the directory where the code was invoked.

## 18.2 File Definitions

The HTWOS\_TDP code reads three or more input files and writes four output files.

### 18.2.1 Control Data File

Control information is read from a file that always has the name “HTWOS\_TDP.INP”. The structure of the file is described as follows, and an example file is provided in Table 18.3:

**Table 18.2** Structure of HTWOS\_TDP.INP Control Data File

Line Numbers	Descriptions
Line 1	Number of radionuclides (N) to extract from the HTWOS annual files
Lines 2 through N+1	For each radionuclide: radionuclide name as listed in the HTWOS file and analyte name to be used for output
Line N+2	Number of chemicals (M) to extract from the HTWOS annual files
Lines N+3 through N+M+2	For each chemical: chemical name as listed in the HTWOS file and analyte name to be used for output
Line N + M + 3	Start year of processing and end year of processing of tank waste
Line N + M + 4	Decay reference year for HTWOS data (based on beginning of year)
Line N + M + 5	Number of waste streams (S) to be defined (corresponding to streams in HTWOS input files)
Lines N+M+6 through N+M+S+5	For each waste stream, one line identifies: 1) stream name as provided in the HTWOS input files 2) stream name to be used in processor output file (for input to the inventory database) 3) waste type to be assigned to this stream in the processor output file.

Line Numbers	Descriptions
Line N+M+S+6	Number of years (Y) of ETF volume data to be read
Lines N+M+S+7 through N+M+S+Y+6	Disposal year associated with each ETF volume set, the volume of ETF liquid disposed of per year (non-WTP sources), and the volume of ETF solid disposed of per year (non-WTP sources)
Line N+M+S+Y+7 (last line)	Flags to indicate if: 1) uranium (U233, U234, U235, U236, and U238) or 2) plutonium (Pu239 and Pu240) isotope amounts are to be added. A flag set to “T” indicates that activities are to be added. A flag set to “F” inhibits addition.

**Table 18.3** Example HTWOS\_TDP.INP File for HTWOS\_TDP

```

2,number of radionuclides,,,,,,
3H,H3,,,,,,
14C,C14,,,,,,
0,Number,of,chemicals,,,,,
2003,2030,Start,and,end,years,,,
2004,Decay,referenced,date,for,HTWOS,input,files,
22,,,
HTWOS-SPENT-RESIN,H2-SPENT-RESIN,Cement,,,
LAW-HEPA1,H2-LAW-HEPA1,cement,,
LAW-HEPA2,H2-LAW-HEPA2,cement,,
LAW-VOC-SCRUB,H2-LAW-VOC-SCRUB,cement,,
HLW-HEPA1,H2-HLW-HEPA1,cement,,
HLW-HEPA2,H2-HLW-HEPA2,cement,,
HLW-VOC-SCRUB,H2-HLW-VOC-SCRUB,cement,,
HLW-AG-MORDENITE-COL,H2-AG-M-COL,cement,,
CONTACT-SUPP-TRU-PACKAGES,H2-CHTRU-PKGS,store,,
REMOTE-SUPP-TRU-PACKAGES,H2-RHTRU-PKGS,store,,
ETF-SOLID-EFFLUENT,H2-ETF-SOLID,cement,,
ETF-LIQUID-EFFLUENT,H2-ETF-LIQUID,liquid,,
BV-PRODUCT,H2-BV-PRODUCT,glass,,
HTWOS-HLW-CANISTERS,H2-HLW-CANISTER,glass,,
FAILED-HLW-MELTERS,H2-HLW-MELTERS,glass,,
FAILED-LAW-MELTERS,H2-LAW-MELTERS,glass,,
HTWOS-LAW-CANISTERS,H2-LAW-CANISTER,glass,,
BV-STACK,H2-BV-STACK,gas,,
WTP-STACK,H2-WTP-STACK,gas,,
HTWOS-LAW-OFFGAS-STACK,H2-LAW-STACK,gas,,
SUPP-TRU-STACK,H2-TRU-STACK,gas,,
242-A-STACK,H2-242A-STACK,gas,,
F,F,! ,addition,flags,for,U,and,Pu

```

### 18.2.2 HTWOS Annual Data Files

The HTWOS annual data files contain information on 22 waste and product streams that form the basis for the waste stream quantities. These HTWOS product streams are listed in Table 18.1. There is a separate file for each year. The range of years is defined in the control data file (line 5 of the example file

shown in Table 18.3). The name of the file for calendar year XXXX is “HTWOXXXX.CSV”. For example, the file for 2004 is named “HTWO2004.CSV”.

Each of the HTWOS annual files have an identical structure and use a comma-separated variables format. They each contain the following information:

**Table 18.4** Structure of the HTWOS Annual Data File

Line Numbers	Description
Lines 1–3	Title information
Line 4	Blank
Line 5	Super headings (radionuclides, chemicals, glass oxides)
Line 6	Column headings (mostly string values)
Lines 7–10	Data for the first tank for the current year
Lines 11–14	Data for the second tank for the current year
...	
Lines 711–714	Data for the last tank for the current year
Lines 715–718	Data for first of 22 waste or product streams
...	
Lines 799–812	Data for last of 22 waste or product streams

Each of lines 7 through 812 has data for 46 radionuclides, 160 chemicals, followed by the volume (gallons), density (g/cc), and mass (kg) of the waste in the tank, product stream, or waste stream. The data for all tanks and streams have four lines. The first line gives the quantity in the “SOLID” phase; the next line gives the quantity in the “LIQUID” phase; the third gives the quantity in the “GAS” phase; and the last line gives the “TOTAL” for the product stream. The HTWOS processor used only the “TOTAL” lines in the analysis.

### 18.2.3 Radionuclide Decay Chain Library

A third input file is a library of radionuclide decay chain data. This file always has the name “RMDLIB.DAT”. The format of this file is not described further because the user typically does not modify this file.

## 18.3 Output Files

The HTWOS\_TDP code writes a log file and an error-message file for each run of the code. The log file is always named “HTWOS\_TDP.OUT” and the error-message file is always named “HTWOS\_TDP.ERR”. These files should be examined after every run to determine whether any unexpected data problems were encountered.

The primary output file is always named “HTWOS\_TDP.CSV” and contains disposal data that is to be imported into the inventory database. This file uses the comma-separated variables format and contains decay-corrected tank leak data. The values generated are the volume of each annual waste stream and the

amount of each analyte in the waste stream. Volumes have units of m<sup>3</sup>; amounts of radionuclides have units of Ci; and amounts of chemicals have units of kg. The tank residual waste is defined to be the amount of material in the tanks as given in the last HTWOS file. Because the values for the product and waste streams in the HTWOS files are cumulative values (increasing with time), the difference of the values for two successive years gives the amount processed during the year. The product and waste streams generated are assumed to be disposed of during the year of processing.

The format of the primary output file is as follows:

- The first line is a comment line.
- The second line contains column titles as follows: Dataset, Waste Stream, Attribute, Year, Code, Analyte (or Volume), Value, Units.
- The remaining lines contain disposal action amounts or volumes with entries corresponding to the titles listed above. The waste streams from the HTWOS input files are given first, starting with the earliest years of processing and ending with the last year of processing. The data for tank residuals is given at the end of the output file, all with the designated last year of processing.

The first 15 lines from an example “HTWOS\_TPD.CSV” file is provided in Table 18.5. A production data set yields an output file containing about 5700 lines of data.

**Table 18.5** Excerpts from the HTWOS\_TDP.CSV File written by HTWOS\_TDP

! HTWOS processing data for tanks and HLW/ILAW after 2000							
"Dataset",	"Waste Stream",	"Attribute",	"Year",	"Code",	"Analyte",	"Value",	"Units",
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Volume,	3.494E-03,	m3,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	I129,	7.370E-09,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Cs137,	1.192E-07,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	C14,	2.080E-06,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Eu152,	1.140E-06,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	U233,	5.440E-08,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	U234,	4.290E-08,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	U235,	1.660E-09,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Np237,	3.910E-11,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	U238,	3.240E-08,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Pu239,	4.930E-11,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	Pu240,	3.790E-08,	Ci,
HTWOS,H2-242A-STACK	,gas		,2004,	mean,	H3,	9.738E-02,	Ci,

The HTWOS\_TDP code writes an additional data file named “STREAMS.OUT”. This file should be ignored.





## 19.0 HTWOS\_TLDP – Hanford Tank Leak Data Processing

### 19.1 Overview

The HTWOS\_TLDP utility code extracts tank leak data from the Hanford Tank Waste Operations Simulator (HTWOS) data files and generates a reformatted data file that is ready for import into the inventory database.

#### 19.1.1 Location in the Processing Sequence

The HTWOS\_TLDP code provides data files to be loaded into the inventory database. It must be run before the inventory code can be run.

#### 19.1.2 How the Code Is Invoked

HTWOS\_TLDP runs in a DOS box under the Windows operating system. A run of HTWOS\_TLDP is initiated by entering the following command line:

```
HTWOS_TLDP
```

The HTWOS\_TLDP utility code does not require any additional command line inputs. The assumption is that all files used by the code will reside in the directory where the code was invoked.

#### 19.1.3 Processing Assumptions

The following assumptions are used in the data processing steps:

- Data values in the HTWOS file have units of Ci for radionuclides and kg for chemicals.
- The total volume of liquid waste leaked is as provided in the HTWOS tank leak file for each tank.
- The leak occurs during the year specified in the HTWOS file.
- The activity value is the amount decayed to the end of the indicated year.

### 19.2 File Definitions

The HTWOS\_TLDP code reads two input files and writes six output files. Only one of the output files (the one named “HTWOS\_LEAK.CSV”) is used in further data processing.

#### 19.2.1 Input Files

Two files are read by the Tank Leak Data Processor code: a control data file and an HTWOS tank leak data file. Both files are text files and the HTWOS tank leak file uses comma-separated variable format.

The control data file defines the number and names of radionuclides to be included and the number and names of chemicals to be included. This file is always named “HTWOS\_LIST.INP”, and an example file is shown in Table 19.1. This file lists the analyte names in two forms: the form provided in the HTWOS tank leak data file (without hyphens) and the form to be written to the disposal action output file. Next the file provides similar information for chemicals to be included in the analysis. This is followed by the decay reference year. The last line read has two logical flags for addition of uranium isotopes and for addition of plutonium-240 to plutonium-239.

**Table 19.1** Example File HTWOS\_LIST.INP for HTWOS\_TLDP

```

20                                ! number of radionuclides
3H      H3,  12.35                ! input name, output name, half life (years)
14C     C14   5730.
79Se    Se79,  0.0
90Sr    Sr90,  29.12
99Tc    Tc99,  0.0
129I    I129,  0.0
137Cs   Cs137, 30.0
152Eu   Eu152, 13.33
231Pa   Pa231 0.0
226Ra   Ra226 1600.
233U    U233,  0.0
234U    U234,  0.0
235U    U235,  0.0
236U    U236,  0.0
238U    U238,  0.0
237Np   Np237, 0.0
239Pu   Pu239,  0.0
240Pu   Pu240, 6537.
241Pu   Pu241,  14.4
241Am   Am241, 432.58
2                                ! number of chemicals
Cr(total  CrVI, 0.0 ! input name, output name, chemical halflife 0.0
U(total)   U, 0.0  ! uranium
2001.      ! reference year for decay ("2000" = Jan. 01, 2001)
F F

```

The column headings in the HTWOS data file are compared to the names of analytes selected in the first file. Data for analytes matching the column headings are written to the output file. The leak amounts are treated as point values.

The second input file contains disposition data (the amounts of analytes leaked and the year leaked for each of the tanks). This file is always named “HTWOS\_TLDP.CSV”. The first line of the file gives the number of tanks described in the file. The second line gives column headings for each field on the following lines. The third line contains the tank ID and analyte IDs for all analytes in the data file. The remainder of the data file contains three lines of data per tank. Each of the remaining lines contains the following information.

- Tank ID: Short title for tank, to be used as waste stream ID (string, ten characters maximum)
- Waste form: “Solid”, “liquid”, or “total”
- For each analyte: amount disposed during the leak event
- Tank\_Year: Year that each tank leaks (integer, four digits).

Only the total values are passed on to the inventory database. No example is provided for this file because formatting the production data set containing data for 148 tanks and 199 analytes for viewing in a text document was impractical.

## 19.2.2 Output Files

The HTWOS\_TLDP code writes a log file and an error-message file for each run of the code. The log file is always named “HTWOS\_TLDP.OUT” and the error-message file is always named “HTWOS\_TLDP.ERR”. These files should be examined after every run to determine whether any unexpected data problems were encountered.

The primary output file is always named “HTWOS\_LEAK.CSV” and contains disposal data that is to be imported into the inventory database. This file uses the comma-separated variables format and contains decay-corrected tank leak data. The first line of the file contains the headings for each column of data. The remaining lines contain the data set name (HTWOS), the tank name, waste type “liquid”, the year of disposal, volume or analyte name, and the amount (volume, activity (Ci), or mass (kg) of each analyte). The first 15 lines of an HTWOS\_LEAK.CSV file are shown in Table 19.2.

**Table 19.2** Excerpts from the HTWOS\_LEAK.CSV File

Source	SiteCode	Subsite	Year	Constituent_name	Constituent_value	Dataset
HTWOS	241-A-101	tle	2015	Volume	3.03E+01	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	I129	2.00E-03	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Cs137	1.68E+03	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	C14	9.78E-02	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Eu152	4.58E-04	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Ra226	1.92E-09	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Pa231	1.56E-06	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	U233	4.65E-05	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	U234	2.59E-05	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	U235	1.10E-06	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	U236	6.17E-07	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Np237	1.00E-03	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	U238	2.42E-05	HTWOS_LEAK_7-04
HTWOS	241-A-101	tle	2015	Pu239	1.60E-02	HTWOS_LEAK_7-04

The HTWOS\_TLDP code writes two additional data files. One is always named “DISP\_TLDP.OUT”. This data file should be ignored. The other output file is always named “TANKLIST.OUT” and contains tank names and leak volumes (gal). The data in these two files are not used in further processing.



## **20.0 Imp\_Med – Median Values for Impact Codes**

### **20.1 Overview**

A run of the ECEM, HUMAN, or TCERM codes with stochastic parameters set to the median value (50th percentile) of their range requires an input keyword file designed for a single realization. The input keyword files developed for these codes typically contain full stochastic distributions for the input variables. A sequence of computer codes is available that reads a full stochastic keyword file and writes the associated median-values keyword file. The same sequence of calculations applies to all three impacts codes.

#### **20.1.1 Location in the Processing Sequence**

Conversion of the stochastic keyword files into median-values keyword files must occur after all environmental release and transport codes have completed but before the impact code is executed.

#### **20.1.2 How the Code Is Invoked**

A sequence of three codes—IMP\_STEP1, STOCHASTIC, and IMP\_STEP2—is used to convert the stochastic keyword file into a median-values keyword file. For purposes of discussion, let the stochastic keyword file be named “stochastic.key” and let the median-values keyword file be named “median.key”. The sequence of three codes is invoked as follows:

```
imp_step1.exe "stochastic.key"  
stochastic.exe "Step1.Key"  
imp_step2.exe "stochastic.key" "median.key"
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

#### **20.1.3 Memory Requirements**

IMP\_STEP1 and IMP\_STEP2 have minimal memory requirements.

### **20.2 Computational Sequence**

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. Invocation of the three codes is described in the following sections.

#### **20.2.1 IMP\_STEP1 Execution**

The first step in the processing sequence is to run the IMP\_STEP1 code. This code reads the stochastic keyword file and writes a separate keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 2000 or XP), IMP\_STEP1 executes in a DOS box. A run of IMP\_STEP1 is initiated by entering the following command line:

```
IMP_STEP1 "stochastic.key"
```

Under the Linux operating system IMP\_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
imp_step1.exe "stochastic.key"
```

For these commands, “IMP\_STEP1” or “imp\_step1.exe” is the name of the executable program and “stochastic.key” is the name of the stochastic keyword file.

### 20.2.2 STOCHASTIC Execution

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the IMP\_STEP1 code and writes a file named “Step1.Val” for use in the IMP\_STEP2 code.

Under the Windows operating system (Releases 2000 or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC "Step1.Key"
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe "Step1.Key"
```

For these commands, “STOCHASTIC” or “stochastic.exe” is the name of the executable program, “Step1.Key” is the name of the input keyword file, and “Step1.Val” is the name of the output file of median values.

### 20.2.3 IMP\_STEP2 Execution

The final step in the processing sequence is to run the IMP\_STEP2 code. This code reads the stochastic keyword file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values keyword file.

Under the Windows operating system (Releases 2000 or XP), IMP\_STEP2 executes in a DOS box. A run of IMP\_STEP2 is initiated by entering the following command line:

```
IMP_STEP2 "stochastic.key" "median.key"
```

Under the Linux operating system IMP\_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
imp_step2.exe "stochastic.key" "median.key"
```

For these commands, “IMP\_STEP2” or “imp\_step2.exe” is the name of the executable program, “stochastic.key” is the name of the stochastic keyword file, and “median.key” is the name of the median-values keyword file.

## **21.0 INGRAB – Inventory Data Extraction**

### **21.1 Overview**

This program reads results (inventory.all file) generated by the SAC Rev. 1 inventory code and outputs files of inventory disposal actions, decay-corrected accumulated inventory, and waste stream volumes by waste type for all waste sites for multiple analytes.

#### **21.1.1 Location in the Processing Sequence**

INGRAB reads a file output by the inventory code. It can be run anytime after the inventory code completes.

#### **21.1.2 How the Code Is Invoked**

INGRAB can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INGRAB executes in a DOS box. A run of INGRAB is initiated by entering the following command line:

```
INGRAB "Keyfilename"
```

Under the Linux operating system, INGRAB is executed through any of the following Bourne Shell or C Shell commands:

```
ingrab-1.exe "Keyfilename"
```

For these commands, “INGRAB.EXE” or “ingrab-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INGRAB is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INGRAB cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **21.1.3 Memory Requirements**

The INGRAB program can require a large amount of memory. Dynamic memory allocation is used and memory use depends on the number of realizations, number of waste sites, and number of waste streams. If the memory requirements exceed approximately 1 gigabyte, the data are subdivided and a number of intermediate scratch files are used. A run of the code using 1042 waste sites, 43 waste streams, and 100 realizations required on the order of 1.2 Gb of memory and 35 Gb of scratch disk space.

### **21.2 File Definitions**

The INGRAB program reads two input files and writes three or four output files. These files are described in the following sections.

### 21.2.1 Input Files

The INGRAB program reads a control keyword file and a results file written by the INVENTORY code that contains all disposal actions (nominally named “inventory.all”). The INGRAB keyword file contains control information. An example file is provided in Table 21.1. Detailed definitions of the keywords are provided in Section 21.3.

**Table 21.1** Example Keyword File for INGRAB

```
REPORT "Ingrab_C14.rpt"
TITLE "Extract C14 for the CA data set on 14 Jun 2004"
USER "Paul W. Eslinger"
FILE ALL      "/home/ANALYSIS4/CA1_median/inv/inventory.all"
FILE AMOUNT "Ingrab_C14.csv"
FILE ACCUM  "IngrabA_C14.csv"
FILE VOLUME "IngrabV_C14.csv"
REALIZAT ALL
ANALYTE      "C14"
HALFLIFE     5.715000E+03
SITE "116-B-1" "116-B-2" PRINT
TYPE ALL PRINT
YEARS SUM 1944 2070
SHOW POSITIVE
END
```

### 21.2.2 Output Files

The INGRAB program writes three or four output files for every run of the code. The output files are the following:

- **Report File.** The report file contains information about the run progress and any error messages. It should be examined after every run of the code.
- **Inventory File.** The inventory file is a text file in comma-separated format that contains the amount disposed for every disposal action (every waste type and every location). This file is organized in chronological order and also contains summary information for all sites and all waste types for each year.
- **Accumulated Inventory File.** The accumulated inventory file is a text file in comma-separated format that contains the accumulated (and decay-corrected) amounts for every disposal action at every site. This file is organized in chronological order and also contains summary information for all sites and all waste types for each year. This file is optional (see the FILE and YEARS keyword descriptions).
- **Volume File.** The volume file is a text file in comma-separated format that contains the volume of waste disposed for every disposal action (every waste type and every location). This file is organized in chronological order and also contains summary information for all sites for each year.



## 21.3 Keyword Definitions INGRAB

In general, the keywords for the INGRAB code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 21.3.1 ANALYTE Keyword for INGRAB

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ "quote 1" ]
```

The single quote string associated must contain an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. An example of this keyword is the following:

```
ANALYTE "H3"
```

### 21.3.2 END Keyword for INGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 21.3.3 EXECUTE Keyword for INGRAB

The optional EXECUTE keyword indicates that the problem defined in the control keyword file should actually be executed. The following is this keyword record's syntax:

```
EXECUTE
```

If the EXECUTE keyword is not entered, the inputs will be checked for errors but no extractions will be performed.

### 21.3.4 FILE Keyword for INGRAB

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [ ACCUM="quote1" ] { ALL="quote2" } { AMOUNT="quote3" } { VOLUME="quote4" }
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the summary inventory file written by the inventory code is associated with the modifier ALL. This file is an input file to INGRAB. The name of the accumulated (and decay-corrected) disposal amounts file written by INGRES is associated with the modifier ACCUM. The name of the disposal amounts file written by INGRES is associated with the modifier AMOUNT. The name of the disposal volumes file written by INGRES is associated with the modifier VOLUME. Example file keywords that define these four files are the following:

```
FILE ALL      "/home/ANALYSIS4/CA1_median/inventory/inventory.all"  
FILE AMOUNT  "Ingrab_C14.csv"  
FILE ACCUM   "IngrabA_C14.csv"  
FILE VOLUME  "IngrabV_C14.csv"
```

### 21.3.5 HALFLIFE Keyword for INGRAB

The HALFLIFE keyword is used to define the halflife of the analyte for which data will be processed. The following is this keyword's syntax:

```
HALFLIFE [N1]
```

The single numerical value must contain the halflife (units are years) of the analyte being processed. The value of 0 should be used if the analyte is not radioactive. An example of this keyword is the following:

```
HALFLIFE 5.6262E-02
```

### 21.3.6 REALIZAT Keyword for INGRAB

The REALIZAT (or REALIZATION) keyword defines the realizations to be used in the inventory data extractions. The following is this keyword's syntax:

```
REALIZAT [ALL | N1]
```

If the modifier ALL is present all realizations, of data will be processed. If the modifier ALL is not present, a single realization number must be entered and the data extraction will occur for only that realization. Two example keywords are the following:

```
REALIZAT ALL  
REALIZAT 4
```

The first example uses all realizations in the data set. The second example uses data from only realization number 4.

### 21.3.7 REPORT Keyword for INGRAB

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 21.3.8 SHOW Keyword for INGRAB

The optional SHOW keyword is used to limit data output to values greater than zero. The following is this keyword's syntax:

```
SHOW POSITIVE
```

The single modifier POSITIVE must be used if the output is to be limited to data values greater than zero. An example of this keyword is the following:

```
SHOW POSITIVE
```

It is recommended that this keyword be used with large data sets. If not, the possibility exists that the output file sizes can exceed the maximum size allowed by the compiler and the run will error terminate.

### 21.3.9 SITE Keyword for INGRAB

The SITE keyword is used to define the sites included in the data extractions. The following is this keyword's syntax:

```
SITE {PRINT} [ALL | "quote1" ... "quoteN"]
```

If the optional PRINT modifier is present, then details of all actions will be printed as well as summary disposal actions. Only summary disposal actions will be printed when PRINT is not present. The choice of which sites to include has two options. The first option is to enter the single modifier ALL, thereby invoking outputs for every waste site with disposal data. The other option is to enter a list of site IDs.

The first example keyword given below uses all sites and will output detailed information for every site. The second example uses only three sites and also outputs detailed information for just these three sites.

```
SITE ALL PRINT  
SITE "116-B-1" "116-B-2" "116-B-3" "116-B-5" PRINT
```

### 21.3.10 TITLE Keyword for INGRAB

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the INGRAB code."
```

### 21.3.11 TYPE Keyword for INGRAB

The TYPE keyword is used to define the waste types included in the data extractions. The following is this keyword's syntax:

```
TYPE {PRINT} [ALL | "quote1" ... "quoteN"]
```

If the optional PRINT modifier is present then details of all actions will be printed as well as summary disposal actions. Only summary disposal actions will be printed when PRINT is not present. The choice of which waste types to include has two options. The first option is to enter the single modifier ALL, thereby invoking outputs for every waste type with disposal data. The other option is to enter a list of waste IDs.

The first example keyword given below uses all waste types and will output detailed information for every waste type. The second example uses only six specific waste types and also outputs detailed information for these six waste types.

```
TYPE ALL PRINT  
TYPE "liquid" "glass" "soil" "cake" "gases" "cement" PRINT
```

### 21.3.12 USER Keyword for INGRAB

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 21.3.13 YEARS Keyword for INGRAB

The YEARS keyword supplies data controlling the time period for the data extraction and whether the accumulated inventory file will be written. The following is this keyword's syntax:

```
YEARS {SUM} [N1 N2]
```

Presence of the optional modifier SUM initiates accumulation calculations that will be output in the accumulation inventory file. The two numerical values, N1 and N2, give the year range (inclusive of the end years) for all of the extraction calculations. The second year cannot be smaller than the first year. The first year must be 1944 or larger. An example keyword entry is the following:

```
YEARS SUM 1944 2070
```

## **22.0 INGRES – Inventory Data Extraction**

### **22.1 Overview**

This program reads an inventory results (\*.res) file generated by the SAC Rev. 1 inventory code and outputs a file of decay-corrected accumulated inventory and waste stream volumes by waste type for all waste sites for a single analyte. All functions performed by this program can also be performed by the INGRAB code. However, INGRAB reads the file “inventory.all” rather than a specific results file and can extract data for more than one realization at a time.

#### **22.1.1 Location in the Processing Sequence**

INGRES reads a file output by the inventory code. It can be run anytime after the inventory code completes.

#### **22.1.2 How the Code Is Invoked**

INGRES can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INGRES executes in a DOS box. A run of INGRES is initiated by entering the following command line:

```
INGRES "Keyfilename"
```

Under the Linux operating system INGRES is executed through any of the following Bourne Shell or C Shell commands:

```
ingres-1.exe "Keyfilename"
```

For these commands, “INGRES.EXE” or “ingres-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INGRES is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INGRES cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **22.1.3 Memory Requirements**

The memory used by INGRES program depends on the number of sites and waste streams. A run of the code using 1000 waste sites and 40 waste streams required on the order of 126 MB of memory.

### **22.2 File Definitions**

The INGRES program reads two input files and writes two output files. These files are described in the following sections.

## 22.2.1 Input Files

The INGRES program reads a control keyword file and a results file written by the INVENTORY code. The INGRES keyword file contains control information. An example file is provided in Table 22.1. Detailed definitions of the keywords are provided in Section 22.3.

**Table 22.1** Example Keyword File for INGRES

```
! Keyword file for INGRES
REPORT "Test.Rpt"
TITLE "Inventory Sums by Site for Testing IngRes"
USER "Paul W. Eslinger"
! The *.res file produced by the inventory code
FILE RES "invl.res"
! The decay corrected accumulated inventory and volume data
FILE SUM "Test_sum.csv"
! Analyte specific data
ANALYTE ID="H3" LAMBDA = 5.6262E-02
! Year for output data
YEAR DECAY=2060 START=1944 STOP=2004
! Waste streams to use in the analysis
WASTE ALL
END
```

## 22.2.2 Output Files

The INGRES program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the information about the inventory assigned to a set of locations decayed to a common year. The first 24 records from an INGRAB output file are provided in Table 22.2. The last block of records in this file (not shown in the table) is the volume of the inventory by waste stream at every waste site.

**Table 22.2** Excerpts from an INGRES Results File

```
"Release by site"
"218-E-ILAW", 0.00000E+00
"218-E-Melter", 0.00000E+00
"atmosphere", 0.00000E+00
"store", 0.00000E+00
"216-U-1&2-Fast", 0.00000E+00
"100-B-15", 0.00000E+00
"100-B-3", 0.00000E+00
"100-B-5", 3.66629E-06
"100-B-8", 0.00000E+00
"100-C-3", 0.00000E+00
"100-C-6", 0.00000E+00
"100-D-23", 0.00000E+00
"100-D-24", 0.00000E+00
"100-D-29", 0.00000E+00
"100-D-3", 4.34891E-03
"100-D-32", 4.28570E-03
"100-D-40", 4.34891E-03
"100-D-42", 0.00000E+00
"100-D-43", 0.00000E+00
```

"100-D-45", 0.00000E+00
"100-D-47", 4.34891E-03
"100-D-49", 0.00000E+00
"100-D-53", 0.00000E+00

## 22.3 Keyword Definitions INGRES

In general, the keywords for the INGRES code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 22.3.1 ANALYTE Keyword for INGRES

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ ID="quote 1" ] [ LAMBDA=N1 ]
```

The single quote string associated with the modifier ID must be an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. The numerical value associated with the modifier LAMBDA is the decay constant (units of year<sup>-1</sup>) for the analyte. Use a value of 0 if the analyte is not radioactive. An example of this keyword is the following:

```
ANALYTE ID="H3" LAMBDA = 5.6262E-02
```

### 22.3.2 END Keyword for INGRES

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 22.3.3 FILE Keyword for INGRES

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the results (\*.res) file written by the inventory code is associated with the modifier RES. The name of the output file from INGRES is associated with the modifier SUM. Example file keywords that define these two files are the following:

```
FILE RES "/home/ANALYSIS4/CA1_median/inventory/inv01.res"
```

```
FILE SUM "Decayed_inventory.csv"
```

### 22.3.4 REPORT Keyword for INGRES

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 22.3.5 TITLE Keyword for INGRES

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the INGRES code."
```

### 22.3.6 USER Keyword for INGRES

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 22.3.7 VERBOSE Keyword for INGRES

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```



### 22.3.8 WASTE Keyword for INGRES

The WASTE keyword is used to define the waste streams for which data summations will be performed. The following is this keyword's syntax:

```
WASTE [ ALL | "quote 1" ... "quote N" ]
```

The single modifier ALL should be entered only if all waste streams are desired. A specific set of waste streams can be identified by entering their identifiers in quote strings. Two examples of this keyword are provided here. The first example performs data summations for all waste streams. The second example performs data summations for 21 specific waste streams.

```
WASTE ALL  
WASTE "soil" "glass" "SF" "soillf" "soilln" "soil3f" "soil3n"  
      "cmntcs", "cmntg3", "rxcomp", "soilsf" "soilsn" "TRASH" "cmntct"  
      "TRUF" "TRUn" "cmntcu" "soiltu" "cmntHT" "cmntH2" "liquid"
```

### 22.3.9 YEAR Keyword for INGRES

The YEAR keyword is used to define the years for specific data actions. The following is this keyword's syntax:

```
YEAR [ START="N1" ] [ STOP=N2 ] [ DECAY=N3 ]
```

The numerical value associated with the modifier START identifies the (integer) calendar year for the start of the data sums. The numerical value associated with the modifier STOP identifies the (integer) calendar year for the end of the data sums. The numerical value associated with the modifier DECAY identifies the (integer) calendar year to which all radioactive elements will be decay corrected. An example of this keyword is the following:

```
YEAR DECAY=2060 STOP=2004 START=1944
```



## 23.0 INPROC – Inventory Preprocessing

### 23.1 Overview

The INPROC code reads a file of disposal action data produced by the SAC inventory database. The information in this file is converted into disposal action keywords for use by the SAC Inventory module. The disposal action keywords are defined as statistical distributions. The INPROC code applies statistical distribution rules, waste type mapping rules, and concentration data fill-in rules in the process of developing the disposal action keywords.

#### 23.1.1 Location in the Processing Sequence

The INPROC code reads data files written by the SAC inventory database. INPROC must be executed before the INVENTORY code can be executed.

The INPROC program interacts with the inventory module of the SAC computer code through data files. In general, the inventory model obtains input data from the following sources:

- Control information from the Keyword Control file and the Environmental Settings file
- Waste stream aggregation names from the Environmental Settings file
- Waste stream aggregation map from the Inventory keyword control file
- Disposal actions from the Master Waste Stream Disposal Action file
- Waste stream selection from the Waste Stream Selection file.

The INPROC program generates the master waste stream disposal action file and aggregation keyword records that are placed in the INVENTORY keyword control file. The INPROC program obtains information from the SAC inventory database provided in a file prepared specifically for INPROC using database queries.

#### 23.1.2 How the Code Is Invoked

INPROC can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INPROC executes in a DOS box. A run of INPROC is initiated by entering the following command line:

```
INPROC
```

Under the Linux operating system INPROC is executed through any of the following Bourne Shell or C Shell commands:

```
inproc-1.exe
```

For these commands, “INPROC.EXE” or “inproc-1.exe” is the name of the executable program. The name of the executable program may contain path information. INPROC reads a file named “inproc.key” from the local directory where the code is invoked. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control

information describing the run. The code will terminate execution after writing an error message to the standard output device if INPROC cannot open the keyword file.

### 23.1.3 Memory Requirements

The INPROC code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INPROC code will require less than 128 MB of memory.

### 23.1.4 Inventory Estimation Rules

Methods to estimate analyte mass or activities are provided when data are missing. Several methods implemented in INPROC are summarized here:

- **Isotopic Ratio Method.** Isotopic ratio information can be used to estimate quantities for missing radionuclides, such as isotopes of uranium and plutonium. If one isotope is known, then the other isotopes can be estimated.
- **Fuel Ratio Method.** When values for fission products are unknown, they may be estimated from the values of known fission products. In the initial assessment, four fission products were considered for this method: cesium-137, strontium-90, technetium-99, and iodine-129. The factors used to estimate activities were defined as a function of year of definition.
- **Uranium Mass.** Conversion of uranium quantities from mass to activity can be made using specific activity factors and assumed isotopic ratios (as defined above for the isotopic ratio method). Similar conversions can be made for uranium mass if the activity of an isotope is known.

### 23.1.5 Statistical Rules

The output parameters from the INPROC program must be expressed as statistical distributions. Because many of the data in the SAC inventory database are stored in the form of point values, rules must be defined to translate the point values into statistical distributions. The rules are supplied to the INPROC program through keywords (see the STATRULE keyword in Section 23.3.7).

The definition of statistical distribution types is limited by the information available on a given parameter. If only one value is given, then assumptions must be made to define a distribution. For example, the single value could be assumed to represent the mean of a normal distribution for which the standard deviation is defined as some factor times the mean. If minimum and maximum values are known, the distribution could be assumed to be uniform over the interval between the two values. These assumptions are supplied using the STATRULE keyword.

## 23.2 File Definitions

The INPROC code reads three input files and writes four output files. These files are described in the following sections.

### 23.2.1 Input Files

The INPROC code reads a control keyword file that is always named “inproc.key”. In addition it reads a file containing aggregate site mappings and another file containing disposal action data. Excerpts from the control keyword file for INPROC are provided in Table 23.1. Example keyword data are provided for each keyword, however the data provided here do not define a complete or internally consistent keyword file.

**Table 23.1** Excerpts from a Keyword Control File for INPROC

```
!   INPROC Keyword File
TITLE "History2_Best Assessment - All Analytes - Best Estimate Data"
USER "Paul W. Eslinger"
VERBOSE
!
! File definitions
FILE DATA  "inproc-input_2005-11-09_DCR-0024.csv" ! Input - inventory
FILE SITE    "History2_Aggr_Site_DCR-0029a.dat"  ! Input - sites
!
FILE AGGREGAT "History2_Best.agg" ! Output file - Used in INVENTORY
FILE WASTE    "History2_Best.daf" ! Output file - Used in INVENTORY
FILE SUMMARY  "History2_Best.rul" ! Output data rule file - Used in INVENT
!
MATCH YEARS
PERIOD START=1944 STOP=12050 CLOSURE=2070
!
! Analyte definitions
ANALYTE ID "U233"  NAME "Uranium-233"          TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U234"  NAME "Uranium-234"          TYPE "NR" COMPUTE OUTPUT
!
! Subset of the subsites
! Atmospheric
ATTRIBUTE "A" TYPE "gas" "Direct atmospheric release"
ATTRIBUTE "gas" TYPE "gas" "Direct atmospheric release"
! Liquid wastes - General
ATTRIBUTE "D" TYPE "river" "River discharge, outfall pipes"
ATTRIBUTE "liquid" TYPE "liquid" "Liquid discharge to ground"
ATTRIBUTE "NR" TYPE "river" "River discharge, N-Reactor cooling water"
ATTRIBUTE "upr" TYPE "liquid" "Unplanned liquid releases"
! Tank wastes
ATTRIBUTE "RES" TYPE "cement" "Waste inside tanks (residuals)"
ATTRIBUTE "TLE" TYPE "liquid" "Liquid releases from tanks"
! Waste to remain at Hanford
ATTRIBUTE "183" TYPE "cement" "183-H waste"
ATTRIBUTE "cement" TYPE "cement" "Miscellaneous cemented wastes"
ATTRIBUTE "CS" TYPE "soil" "Commercial, Segregated (for US Ecology site)"
ATTRIBUTE "CU" TYPE "soil" "Commercial, Unsegregated (US Ecology site)"
ATTRIBUTE "glass" TYPE "glass" "Glass waste forms (ILAW and HLW Glass)"
ATTRIBUTE "LCN" TYPE "cement" "Lithium Targets in Caissons at 618-11"
ATTRIBUTE "LTN" TYPE "soil" "Lithium Targets in Trenches at 618-11"
ATTRIBUTE "NC" TYPE "cement" "Non-encapsulated waste from WESF"
ATTRIBUTE "RSB" TYPE "soil" "Reactor Storage Basin (differentiated from
reactor structure)"
```

```
ATTRIBUTE "S" TYPE "core" "Reactor core"
ATTRIBUTE "S1F" TYPE "soillf" "Segregated Trench Category 1 Waste"
!
! Subset of the waste types
WASTYPE NAME "cmntsn"
WASTYPE NAME "cmntuf"
WASTYPE NAME "core"
WASTYPE NAME "gas"
WASTYPE NAME "glass"
WASTYPE NAME "HLW"
WASTYPE NAME "liquid"
WASTYPE NAME "NM1"
WASTYPE NAME "NM2"
!
! Statistical rules
! These rules pick out point values as the best estimates
! The SIM best estimate is the mean value
!
! Paul W. Eslinger : 13 Jan 2006 : Pick out the mean value from SIM data
STATRULE DATASET "SIM8c" MEAN "ANALYTE" PASSTHRU
  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
STATRULE DATASET "SIM8d" MEAN "ANALYTE" PASSTHRU
  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
!
! Paul W. Eslinger : 13 Jan 2006 : The point values (MEAN input data)
STATRULE DATASET "ALL" MEAN  "ANALYTE" APPLY "CONSTANT" 0.0 YEARS ALLTIMES
STATRULE DATASET "ALL" MEAN  "VOLUME"  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
!
! Paul W. Eslinger : 13 Jan 2006 : The point values (POINT input data)
STATRULE DATASET "ALL" POINT "ANALYTE" APPLY "CONSTANT" 0.0 YEARS ALLTIMES
STATRULE DATASET "ALL" POINT "VOLUME"  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
!
! Paul W. Eslinger : 13 Jan 2006 : The point values (MODE input data)
STATRULE DATASET "ALL" MODE  "ANALYTE" APPLY "CONSTANT" 0.0 YEARS ALLTIMES
STATRULE DATASET "ALL" MODE  "VOLUME"  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
!
! Paul W. Eslinger : 13 Jan 2006 : Special rule for constant analytes (clean
water concentrations)
STATRULE DATASET "ALL" CONSTANT "ANALYTE" APPLY "CONSTANT" 0.0 YEARS ALLTIMES
STATRULE DATASET "ALL" CONSTANT "VOLUME"  APPLY "CONSTANT" 0.0 YEARS ALLTIMES
!
! Summation rules
SUM TO "U238" FROM 2  "U234" "U238"
!
! Surrogate site rules
SURROGATE UNKSITE="100-B-5" 24150 1954 SURSITE="116-B-1" 1951 "liquid"
SURROGATE UNKSITE="100-B-5" 24150 1955 SURSITE="116-B-1" 1951 "liquid"
SURROGATE UNKSITE="100-B-5" 24150 1956 SURSITE="116-B-1" 1951 "liquid"
SURROGATE UNKSITE="100-D-3" 434.782 1944 SURSITE="118-D-2" 1956 "UTN"
SURROGATE UNKSITE="100-D-3" 434.782 1945 SURSITE="118-D-2" 1956 "UTN"
!
! Estimation factors using U238 as the known analyte
ESTIMATE ANALYTE "U234" FROM "U238" FACTOR 0.97 YEARS ALLYEARS
  SITES SITEFILE="Estimate_Uranium_IDs_DCR-0029.dat"
```

```
ESTIMATE ANALYTE "U235" FROM "U238" FACTOR 0.042 YEARS ALLYEARS
  SITES SITEFILE="Estimate_Uranium_IDs_DCR-0029.dat"
!
! Estimation factors using U235 as the known analyte
ESTIMATE ANALYTE "U234" FROM "U235" FACTOR 23.21 YEARS ALLYEARS
  SITES SITEFILE="Estimate_Uranium_IDs_DCR-0029.dat"
ESTIMATE ANALYTE "U238" FROM "U235" FACTOR 23.86 YEARS ALLYEARS
  SITES SITEFILE="Estimate_Uranium_IDs_DCR-0029.dat"
!
! Fission product estimation factors using Cs137 as the known analyte
! Table 4.16, Year 1 : All years before 1973
ESTIMATE ANALYTE "Se79" FROM "Cs137" FACTOR 2.476E-7 YEARS BEFORE 1973
  SITES SITEFILE="Estimate_Fission_IDs_DCR-0029.dat"
ESTIMATE ANALYTE "Sr90" FROM "Cs137" FACTOR 9.160E-1 YEARS BEFORE 1973
  SITES SITEFILE="Estimate_Fission_IDs_DCR-0029.dat"
ESTIMATE ANALYTE "Tc99" FROM "Cs137" FACTOR 1.418E-4 YEARS BEFORE 1973
  SITES SITEFILE="Estimate_Fission_IDs_DCR-0029.dat"
ESTIMATE ANALYTE "I129" FROM "Cs137" FACTOR 3.041E-7 YEARS BEFORE 1973
  SITES SITEFILE="Estimate_Fission_IDs_DCR-0029.dat"
ESTIMATE ANALYTE "Eu152" FROM "Cs137" FACTOR 1.320E-5 YEARS BEFORE 1973
  SITES SITEFILE="Estimate_Fission_IDs_DCR-0029.dat"
!
END
```

A shortened example of an aggregate site mapping file for INPROC is provided in Table 23.2. The first line contains the number of sites and labels for the three columns. INPROC allows aggregation of waste forms from several sites (first entry on the data line) to one output site (second entry on the data line). The site list must be complete with respect to the sites defined in the inventory data file.

**Table 23.2** Aggregate Site Mapping File for INPROC

8,SiteCode,SiteCode
100-B-3,100-B-3
100-B-5,100-B-5
100-B-8,100-B-8
100-B-15,100-B-15
100-C-3,100-C-3
100-C-6,100-C-6
100-D-3,100-D-3
100-D-23,100-D-23

A shortened version of the inventory disposal data is provided in Table 23.3. This example file contains data for three sites (100-H-5, 100-N-66, and 200-E-100) for a few years. The “derived-records” data sets will have statistical rules assigned while the “SIM\_7\_Jun\_04” data set has a user-defined distribution that will be passed through without modification. The production data set for the 2004 Composite Analysis contains over 1.5 million records.

**Table 23.3** Excerpted Records from an INPROC Inventory Action File

"Derived-Records", "100-H-5", "soil", 1953, "Mean", "Cl4", 2.08, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "Cs137", 2.13, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "Eu152", 73.91, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "H3", 0.86, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "Sr90", 0.32, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "U", 1.77857e-04, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "U234", 2.38e-07, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "U235", 2.49e-09, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "U236", 9.78214e-10, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "U238", 5.94042e-08, "dpa4[0]", 1865, "none",
"Derived-Records", "100-H-5", "soil", 1953, "Mean", "Volume", 7345.00, "dpa4[0]", 1865, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Am241", 0.30, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Cl14", 9550.00, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Cl136", 75.00, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Cs137", 40.00, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Eu152", 58.00, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "H3", 64000.00, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Sr90", 14.30, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Tc99", 0.03, "dpaA/2003", 3941, "none",
"Derived-Records", "100-N-66", "S", 1998, "Point", "Volume", 500.00, "dpaA/2003", 3941, "none",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0200", "U238", 3.5874e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0050", "U238", 1.8203e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0450", "U238", 5.3728e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0950", "U238", 1.0543e-09, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0250", "U238", 4.0079e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0100", "U238", 2.5472e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0750", "U238", 7.7905e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0150", "U238", 3.1116e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "Mean", "U238", 5.9278e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0900", "U238", 9.6264e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0800", "U238", 8.3154e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0995", "U238", 1.2114e-09, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0700", "U238", 7.3086e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0650", "U238", 6.874e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0600", "U238", 6.4724e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0550", "U238", 6.0841e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0500", "U238", 5.7285e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0850", "U238", 8.9016e-10, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0400", "volume", 4.14, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0200", "volume", 3.57, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0350", "volume", 4.02, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0300", "volume", 3.88, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0250", "volume", 3.73, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0150", "volume", 3.38, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0100", "volume", 3.16, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0050", "volume", 2.88, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "SD", "volume", 0.89, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0600", "volume", 4.61, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "Mean", "volume", 4.38, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0005", "volume", 2.40, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0995", "volume", 6.35, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0550", "volume", 4.49, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0650", "volume", 4.73, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0700", "volume", 4.87, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0750", "volume", 5.02, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0800", "volume", 5.18, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0850", "volume", 5.37, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0900", "volume", 5.59, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0950", "volume", 5.87, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0450", "volume", 4.26, "UNA200AREA", 4360, "None",
"SIM_7_Jun_04", "200-E-100", "liquid", 1945, "V0500", "volume", 4.38, "UNA200AREA", 4360, "None",

### 23.2.2 Output Files

The INPROC program writes four output files. One file is a report file that contains text information about the run of the code. The report file is always named “inproc.out”. Searching for the text strings “error”, “warning”, or “skip” in the report file is useful for helping identify data set inconsistencies.



The disposal action file contains information about every disposal after all statistical, estimation, or surrogate rules have been applied. The information is given as a statistical distribution for volume and statistical distributions for the concentrations of each analyte. There is a separate DISPOSAL keyword for every combination of waste site, waste stream type, and year of disposal. This file is read and processed in the INVENTORY code. The first few lines of a disposal action file are provided in Table 23.4.

**Table 23.4** Excerpts from a Disposal Action File Written by INPROC

```
TITLE "2004 Composite Analysis Baseline Inventory"
USER "Paul W. Eslinger"
DISPOSAL WASTEID "100-H-5" TYPE "soil" YEAR 1953
VOLUME 6 5.87600E+03 7.34500E+03 8.81400E+03 TRUNCATE 0.000 1.000
CONTAM1 "H3" " 9 -9.85732E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM2 "C14" " 9 -8.97413E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM6 "Sr90" " 9 -1.08459E+01 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM8 "Cs137" " 9 -8.95037E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM9 "Eu152" " 9 -5.40365E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM14 "U235" " 9 -2.95175E+01 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM15 "U238" " 9 -2.47347E+01 1.26864E+00 TRUNCATE 0.010 0.990
DISPOSAL WASTEID "100-K-2" TYPE "soil" YEAR 1955
VOLUME 6 5.87600E+03 7.34500E+03 8.81400E+03 TRUNCATE 0.000 1.000
CONTAM1 "H3" " 9 -9.85732E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM2 "C14" " 9 -8.97413E+00 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM14 "U235" " 9 -2.95175E+01 1.26864E+00 TRUNCATE 0.010 0.990
CONTAM15 "U238" " 9 -2.47347E+01 1.26864E+00 TRUNCATE 0.010 0.990
```

The summary file provides a summary of the application of the estimation and surrogate data rules. This file contains information about every site, waste type at the site, waste volume, analyte, and year of disposal. The first few records from a summary file are provided in Table 23.5. The numerical values are point values for the respective action. Lines of data where an input distribution was supplied has a last entry of “inp”. Lines of data filled in from a surrogate site has a last entry of “sur”. Lines of data filled in using an estimation rule have a last entry of “est”.

**Table 23.5** Excerpts from a Summary Action File Written by INPROC

```
Derived-Records, 100-H-5, soil, 1953, volume, 7.34500E+03, inp
Derived-Records, 100-H-5, soil, 1953, H3, 5.23627E-05, inp
Derived-Records, 100-H-5, soil, 1953, C14, 1.26645E-04, inp
Derived-Records, 100-H-5, soil, 1953, Sr90, 1.94838E-05, inp
Derived-Records, 100-H-5, soil, 1953, Cs137, 1.29689E-04, inp
Derived-Records, 100-H-5, soil, 1953, Eu152, 4.50015E-03, inp
Derived-Records, 100-H-5, soil, 1953, U235, 1.51608E-13, inp
Derived-Records, 100-H-5, soil, 1953, U238, 1.81080E-11, inp
Derived-Records, 100-K-2, soil, 1955, volume, 7.34500E+03, sur
Derived-Records, 100-K-2, soil, 1955, H3, 5.23627E-05, sur
Derived-Records, 100-K-2, soil, 1955, C14, 1.26645E-04, sur
Derived-Records, 100-K-2, soil, 1955, Sr90, 1.94838E-05, sur
Derived-Records, 100-K-2, soil, 1955, Cs137, 1.29689E-04, sur
Derived-Records, 100-K-2, soil, 1955, Eu152, 4.50015E-03, sur
Derived-Records, 100-K-2, soil, 1955, U235, 1.51608E-13, sur
```

The INPROC code produces a suite of WASTEMAP keywords for use in the INVENTORY code. An example of this output file for the first eight WASTEMAP keywords is provided in Table 23.6.

**Table 23.6** Excerpts from an Aggregate Keyword File Written by INPROC

TITLE	"2004 Composite Analysis Baseline Inventory"
USER	"Paul W. Eslinger"
WASTEMAP AGGREGAT	"100-H-5" FRACTION 1 1.0 STREAMS
	"100-H-5 soil 1953"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1955"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1956"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1957"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1958"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1959"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1960"
WASTEMAP AGGREGAT	"100-K-2" FRACTION 1 1.0 STREAMS
	"100-K-2 soil 1961"

## 23.3 Keyword Definitions for INPROC

The main control for program operation is provided in the keyword control file, which always has the name `inproc.key`. Options and parameters are provided in this file using keywords. This section defines the keywords available for the INPROC program.

In general, the keywords for INPROC can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 23.3.1 ANALYTE Keyword for INPROC

The ANALYTE keyword identifies contaminants to be included in the analysis and provides parameter values associated with the contaminant. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"] [TYPE="quote 2"] [NAME="quote 3"]
{COMPUTE} {OUTPUT}
```

The single quote string associated with the modifier ID must contain an analyte identification (limit of six characters in length) from the set of the analytes in the inventory results file. The TYPE modifier is followed by a two-character quote string. The TYPE modifiers associated with the ANALYTE keyword are given in Table 23.7.

**Table 23.7** TYPE Modifiers Associated with the ANALYTE Keyword in INPROC

Modifier	Description
NR	Analyte is non-organic radionuclide.
NS	Analyte is non-organic stable element or compound.
OR	Analyte is organic radionuclide.
OS	Analyte is organic stable element or compound.

The NAME modifier is followed by a descriptive name for the contaminant in a quote string of up to 72 characters. The presence of the optional COMPUTE modifier causes the contaminant to be included in the analysis. As the ANALYTE keyword records are read, the COMPUTE modifier is searched for first. If it is not found, the record is ignored. The presence of the optional modifier OUTPUT causes data for the analyte to be output in the disposal action file. In the following five keyword examples, tritium will be ignored and uranium-234 will be calculated but not output:

```
ANALYTE ID "H3"    NAME "Tritium"    TYPE "NR"
ANALYTE ID "C14"   NAME "Carbon-14"  TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U234"  NAME "Uranium-234" TYPE "NR" COMPUTE
ANALYTE ID "U235"  NAME "Uranium-235" TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U238"  NAME "Uranium-238" TYPE "NR" COMPUTE OUTPUT
```

### 23.3.2 ATTRIBUTE Keyword for INPROC

The ATTRIBUTE keyword provides cross-reference information relating the input attribute field names to the output waste type names. The following is this keyword's syntax:

```
ATTRIBUTE ["quote 1"] [TYPE "quote 2"]
```

The first quote string for this keyword contains the name of an attribute field that may be supplied in the input disposal action file. The quote string associated with the TYPE modifier is the waste type to be assigned to the output data set whenever the input attribute field occurs. The following six examples illustrate the use of this keyword:

```
ATTRIBUTE "STF"    TYPE "soilsf"  "segregated trench offsite waste"
ATTRIBUTE "STN"    TYPE "soilsn"  "segregated trench onsite"
ATTRIBUTE "TCN"    TYPE "cmntct"  "segregated TRU caisson onsite"
ATTRIBUTE "UTN"    TYPE "soiltu"  "unsegregated trench onsite"
ATTRIBUTE "tle"    TYPE "liquid"   "Liquid releases from tanks"
ATTRIBUTE "res"    TYPE "cake"     "Waste inside the tanks"
```

### 23.3.3 END Keyword for INPROC

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 23.3.4 ESTIMATE Keyword for INPROC

The ESTIMATE keyword provides factors to estimate concentrations of analytes not present in a waste disposal action. This feature is used when an analyte is expected to be present but no information is available in the database. The estimate is based on other analytes that are present. The ESTIMATE keyword records can be defined to apply to specific waste sites and years, or they can be applied to all disposal actions. The following is this keyword's syntax:

```
ESTIMATE [ANALYTE "quote1"] [FROM "quote2"] [FACTOR N1]  
[SITES [ SITEFILE "quote3" | ALLSITES | {NOTSITES} "quote4" ... "quoteN"]]  
[YEARS [ALLYEARS|BEFORE N2|AFTER N2|AT N2]]
```

At least one ESTIMATE keyword must be entered, even if it is not used in the calculations. The quote string associated with the ANALYTE modifier identifies the analyte for which concentration data are to be estimated. The quote string associated with the FROM modifier identifies the analyte to be used as the source of the estimate. The numerical value N1 associated with the FACTOR keyword is the factor that the concentration of analyte "quote2" is multiplied by to estimate the concentration of analyte "quote1."

The SITES modifier is used to define the sites where the estimation rule is to be applied. If the modifier ALLSITES is present along with the SITES modifier, then specific site IDs are not used, and the method is applied to all sites needing a fill-in rule. If the modifier SITEFILE is used, a file name is provided for a file that contains a list of site IDs. This file is a text file where one site ID is contained on each line of the file. If neither the ALLSITES modifier nor the SITEFILE modifier is present, the strings "quote4" through "quoteN" provide the site IDs where this estimation rule will be applied. If the modifier NOTSITES is present, but not the modifiers ALLSITES or SITEFILE, then the site names identified in the quote STRINGS are excluded and all other sites are included for application of the ESTIMATE rule.

The years for which the estimation rule applies are provided using the YEARS modifier. If the modifier ALLYEARS is present along with the YEARS modifier, then the factor is applied to all years and the numerical value N2 is not used. Otherwise, specific years of applicability are defined using one of the modifiers BEFORE, AFTER, or AT. When these modifiers are used, the value of N2 is used to define the year or range of years for application of the ESTIMATE rule. When BEFORE is present, the rule is applied to all years before N2. When AFTER is present, the rule is applied to all years after N2. When AT is present, the rule is applied only to year N2. Only one of the BEFORE, AFTER, or AT modifiers can be used (there is an input hierarchy of BEFORE, AFTER, and AT). For example, if BEFORE is present, then AFTER and AT would be ignored if they were present. If no year data are provided, the code will error terminate.

The order of the entry for multiple ESTIMATE records is important. If more than one method is provided for estimation of the same analyte, then the first estimation rule where the necessary information is available is used. For example, if two methods are given for estimation of <sup>99</sup>Tc inventory, the first based on the <sup>90</sup>Sr concentration and the second on the <sup>129</sup>I concentration, the analysis will use the factor based on the <sup>90</sup>Sr concentration (first estimate rule) if a value for <sup>90</sup>Sr concentration is given. If no <sup>90</sup>Sr concentration is given for the waste site, then the code will attempt to estimate the <sup>99</sup>Tc concentration from the <sup>129</sup>I concentration (second estimation rule). If there is no concentration for either <sup>90</sup>Sr or <sup>129</sup>I, then no estimate will be made.

For example, the following five ESTIMATE keywords can be used to estimate fission product concentrations from fuel rod failures in N reactor fuel using <sup>137</sup>Cs as the known analyte:

```
ESTIMATE ANALYTE "Se79" FROM "Cs137" FACTOR 2.476E-7 ALLTIMES
SITES "116-N-1" "116-N-3"
ESTIMATE ANALYTE "Sr90" FROM "Cs137" FACTOR 9.160E-1 ALLTIMES
SITES "116-N-1" "116-N-3"
ESTIMATE ANALYTE "Tc99" FROM "Cs137" FACTOR 1.418E-4 ALLTIMES
SITES "116-N-1" "116-N-3"
ESTIMATE ANALYTE "I129" FROM "Cs137" FACTOR 3.041E-7 ALLTIMES
SITES "116-N-1" "116-N-3"
ESTIMATE ANALYTE "Eu152" FROM "Cs137" FACTOR 1.320E-5 ALLTIMES
SITES "116-N-1" "116-N-3"
```

The following example defines estimation factors for <sup>235</sup>U using <sup>234</sup>U as the known analyte:

```
ESTIMATE ANALYTE "U235" FROM "U234" FACTOR 0.043 YEARS ALLYEARS
SITES ALLSITES
```

The following example illustrates the use of the SITEFILE modifier to specify the sites for which the estimation rule is to be applied:

```
ESTIMATE ANALYTE "Eu152" FROM "Cs137" FACTOR 1.320E-5 YEARS BEFORE 1973
SITES SITEFILE="Fission_IDs.dat"
```

### 23.3.5 FILE Keyword for INPROC

The FILE keyword is used to enter the names of all files except the input keyword file and the report file. Those two files always have the names “inproc.key” and “inproc.out”, respectively. The following is this keyword’s syntax:

```
FILE modifier1 "quote1" ... modifier5 "quote5"
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The modifiers associated with the FILE keyword are given in Table 23.8. The file name associated with a modifier must be entered before the next modifier is entered.

**Table 23.8** Modifiers Associated with the FILE Keyword in INPROC

Modifier	Description of Use
AGGREGAT	Output file containing aggregation keyword data output for use in the inventory code
DATA	Input file containing inventory disposal data (comma-separated format)
SITE	Input file containing waste disposal site aggregation rules and default waste types
SUMMARY	Output file containing point values for all disposal actions and data source indicators
WASTE	Output file containing stochastic definitions of disposal actions for all waste streams, contaminants, and years

At least one FILE keyword is required for every run of the code, and multiple keyword entries can be used. An example entry is the following:

```
FILE DATA=INPROC_Input.csv" SITE="Aggr_site.csv" SUMMARY="CA1-stoch.out"  
AGGREGAT"CA1-aggregat.key" WASTE="CA1-stoch.daf"
```

The following set of five entries has the same effect as the single entry above.

```
FILE DATA      "INPROC_Input.csv"  
FILE SITE       "Aggr_site.csv"  
FILE SUMMARY    "CA1-stoch.out"  
FILE AGGREGAT   "CA1-aggregat.key"  
FILE WASTE      "CA1-stoch.daf"
```

### 23.3.6 PERIOD Keyword for INPROC

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
PERIOD [START=year1] [STOP=year2] [CLOSURE=year3]
```

The modifier START and the value year<sub>1</sub> identify the start of the simulation period. The start of the simulation period must be 1944 or later, or the inventory code will error terminate. The modifier STOP and the value year<sub>2</sub> identify the end of the simulation period. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The modifier CLOSURE and the value year<sub>3</sub> identify the year that site closure occurs. The year of site closure cannot be smaller than the start year. The following is an example PERIOD keyword that simulates from 1944 through 3050 with site closure occurring at 2050:

```
PERIOD START=1944 STOP=3050 CLOSURE=2050
```

### 23.3.7 STATRULE Keyword for INPROC

The STATRULE keyword is used to define statistical distribution rules to convert available parameter information into statistical distributions for output (Table 23.9). The following is this keyword's syntax:

```
STATRULE DATASET "setname" {TRUNCATE P1 P2}  
  [POINT|MEAN|MEDIAN|MODE|CONSTANT|INPUT][ "ANALYTE" | "VOLUME" ]  
  [[APPLY {PASSTHRU}] "CONSTANT" | "NORMAL" SD | "LOGE" SD |  
  "LOG10" SD | "UNIFORM" UMIN UMAX | "DISCRETU" UMIN UMAX |  
  "TRIANGLE" FMIN FMAX | "USER" ]]  
  [[YEARS] [ALLTIMES|BEFORE T1|AFTER T2|AT T3]]
```

Each STATRULE keyword record provides information for one statistics rule. Multiple rules can be defined. The rules are tested for applicability to a data set in the order they occur in the keyword file. If the conditions of the rule are satisfied for data representing a given parameter (a POINT value is given in the correct time range, for example), then the rule is used for the data point.

The quote string associated with the DATASET modifier must contain the name of a data set ID in the inventory disposal action data. Use of this modifier allows specification of rules specific to the data sets in the database. The DATASET modifier is mandatory. If a rule is to be applied to all data sets, then the data set name "all" can be entered.

One of the modifiers CONSTANT, INPUT, MEAN, MEDIAN, MODE, or POINT are used to indicate the type of input distribution for which the rule applies. These rules are applied to the data in the inventory disposal data file. The modifier INPUT is used to direct the code to use the input values without modification. When a data input is found that matches the modifier on the statistics rule, then the rule may be applied if other constraints are met (correct data set and year of application, for example). Each of the modifiers CONSTANT, INPUT, MEAN, MEDIAN, MODE, or POINT must be followed by a quote string containing either “ANALYTE” or “VOLUME” but not both. This quote string identifies whether the rule applies to analyte amounts or to volumes. If PASSTHRU is present, the data are treated as concentrations and are passed through to output without modification.

**Table 23.9** Statistical Distributions on the STATRULE Keyword for INPROC

Modifier	Associated Parameters and Description of Use
CONSTANT	The data in the inventory data disposal file are treated as constants. No statistical distribution is applied to these data records.
UNIFORM	A uniform distribution is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
DISCRETU	A discrete uniform distribution (range of integer values) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
LOGU10	A loguniform distribution (base 10) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
LOGUE	A loguniform distribution (base e) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
TRIANGLE	The single datum in the inventory data disposal file is treated as the mode of a triangular distribution. The minimum of the triangular distribution is the product of FMIN and the mode. The maximum of the triangular distribution is the product of FMAX and the mode.
NORMAL	The single datum in the inventory data disposal file is treated as coming from a normal distribution. The datum provides the mean of the normal distribution. The numerical value SD is applied as the standard deviation of the distribution.
LOG10	The single datum in the inventory data disposal file is treated as coming from a lognormal distribution (base 10). The datum provides the arithmetic mean of the lognormal distribution. The numerical value SD is applied as the arithmetic standard deviation of the distribution.
LOGE	The single datum in the inventory data disposal file is treated as coming from a lognormal distribution (base e). The datum provides the arithmetic mean of the lognormal distribution. The numerical value SD is applied as the arithmetic standard deviation of the distribution.

An optional specification of years for application of the rule is handled with the YEARS modifier. If YEARS is absent, the rule is applied to all years. Use of the YEARS modifier followed by the ALLTIMES modifier also applies the statistics rule for all times. The numerical value (integer year) associated with the optional BEFORE modifier specifies that the statistics rule will be applied to all years

before the year provided. The numerical value (integer year) associated with the optional AT modifier specifies that the statistics rule will be applied to the one year provided. The numerical value (integer year) associated with the optional AFTER modifier specifies that the statistics rule will be applied to all years after the year provided.

If the TRUNCATE modifier is present then, the distribution will have the TRUNCATE option activated with the parameter values P1 and P2 being the lower and upper truncation limits. Truncation limits are defined as tail probabilities rather than limits on the data values. The truncation is only valid for the following distributions: uniform, loguniform, normal, lognormal (base e), lognormal (base 10), and triangular.

Two examples of STATRULE keyword records that apply to volumes are the following:

```
STATRULE DATASET "ALL" POINT "VOLUME" APPLY "TRIANGLE" 0.8 1.2
TIMES ALLTIMES
STATRULE DATASET "ALL" MEAN "ANALYTE" APPLY "LOGE" 2.0 TIMES BEFORE 1970
```

The first keyword indicates that volume data with a statistical parameter name of “point” will be assigned a triangular distribution with the point value being the mode, the minimum will be 0.8 times the point value, and the maximum will be 1.2 times the point value. The rule will be applied to all time periods. The second keyword indicates that volume data with a statistical parameter name of “mean” will be assigned a lognormal (base e) distribution with the input value being used as the arithmetic mean and the arithmetic standard deviation will be set to 2.0. The rule will be applied for times before 1970.

The following two examples apply rules to a specific data set titled “SIM\_7\_Jun\_04”.

```
STATRULE DATASET "SIM_7_Jun_04" INPUT "VOLUME" PASSTHRU
APPLY "USER" 0.80 1.2 YEARS ALLTIMES
STATRULE DATASET "SIM_7_Jun_04" INPUT "ANALYTE" PASSTHRU
APPLY "USER" 0. 0. YEARS ALLTIMES
```

The following two examples apply rules to all data sets for different time periods and also illustrate the use of the truncation option.

```
STATRULE DATASET "ALL" MEAN "ANALYTE" APPLY "LOGE" 2.00
YEARS BEFORE 1970 TRUNCATE .01 .99
STATRULE DATASET "ALL" MEAN "ANALYTE" APPLY "LOGE" 0.25
YEARS AFTER 1969 TRUNCATE .01 .99
```

### 23.3.8 SURROGATE Keyword for INPROC

The SURROGATE keyword provides information on use of surrogate sites to provide analyte concentration estimates for sites having no analyte data. The following is this keyword’s syntax:

```
SURROGATE UNKSITE="quote 1" N1 N2 SURSITE="quote 2" N3 "quote 3"
```

The ID (“quote 1”) of the site for which data are to be generated is associated with the modifier UNKSITE. The two numerical values associated with the modifier UNKSITE are the volume of the waste disposed to the site with missing data (N1, m<sup>3</sup>) and the year the disposal action occurs (N2, calendar year). The ID (“quote 2”) of the surrogate site from which analyte data are to be used is



associated with the modifier SURSITE . The numerical value associated with SURSITE is the year of disposal for the source surrogate data., and “quote 3” is the waste type of the site providing the surrogate data. If the input data file does not contain data for the reference site, waste type, and year, then the unknown site disposal action is not generated and no error message is written.

An example SURROGATE keyword record is as follows:

```
SURROGAT UNKSITE "216-W-32" 1965 30.0 SURSITE "216-W-5" 1964 "liquid"
```

The record indicates data for site 216-W-5 in year 1964 and waste type “liquid” will be used to estimate the disposal action to site 216-W-32 for year 1965, with the disposal volume being 30 m<sup>3</sup>. The waste type for the unknown site is always the same as for the surrogate site (“liquid” in the example).

### 23.3.9 SUM Keyword for INPROC

The SUM keyword allows radionuclide concentration addition such as summing over all uranium isotopes. The following is this keyword’s syntax:

```
SUM [TO "quote 1"] [FROM N1 "quote 2" {"quote 3", ... "quote N1"}]
```

The record specifies the name of the analyte representing the output analyte, and the names of all analytes to be included in the sum.

The quote string associated with the TO modifier contains the ID for the output analyte. The numerical value associated with the FROM modifier indicates the number of analytes to use in the sum. The quote strings associated with the FROM modifier provide the analyte IDs used in the sum. An example record that outputs uranium-238 as the sum of data for uranium-234 and uranium-238 is the following:

```
SUM TO "U238" FROM 2 "U234" "U238"
```

### 23.3.10 VERBOSE Keyword for INPROC

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

If this keyword is present, the report file will contain a listing of the actions by analyte (data distribution, estimated distribution, or analyte not included) for every data set. In addition, all surrogate sites that use a specific waste stream from a source site are identified.



## **24.0 Inv\_Med – Median Values for Inventory**

### **24.1 Overview**

A run of the INVENTORY code with stochastic parameters set to the median value (50th percentile) of their range requires an input disposal action file designed for a single realization. The disposal action file developed by the INPROC computer code contains stochastic distributions for volumes and concentrations and is intended to support multiple realization runs of the INVENTORY code. A sequence of computer codes is available that reads a full stochastic disposal action file and writes the associated median-values disposal action file.

#### **24.1.1 Location in the Processing Sequence**

The disposal action file is an output of the INPROC code and an input to the INVENTORY code. Therefore, conversion of the stochastic disposal action file into a median-values disposal action file must occur after a run of the INPROC code but before a run of the INVENTORY code.

#### **24.1.2 How the Code Is Invoked**

A sequence of three codes—INV\_STEP1, STOCHASTIC, and INV\_STEP2—is used to convert the stochastic disposal action file into a median-values disposal action file. For purposes of discussion, let the stochastic disposal action file be named “stochastic.daf” and let the median-values disposal action file be named “median.daf.” The sequence of three codes is invoked as follows:

```
inv_step1.exe "stochastic.daf"  
stochastic.exe "Step1.Key"  
inv_step2.exe "stochastic.daf" "median.daf"
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

#### **24.1.3 Memory Requirements**

INV\_STEP1 and INV\_STEP2 have minimal memory requirements. “Stochastic.exe” uses 20 MB of memory for a run of 107 years, 988 waste sites, and 40 waste types.

## **24.2 Computational Sequence**

A sequence of three codes is used to convert the stochastic disposal action file into a median-values disposal action file. Invocation of the three codes is described in the following paragraphs.

### **24.2.1 INV\_STEP1 Execution**

The first step in the processing sequence is to run the INV\_STEP1 code. This code reads the stochastic disposal action file and writes a keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 2000 or XP), INV\_STEP1 executes in a DOS box. A run of INV\_STEP1 is initiated by entering the following command line:

```
INV_STEP1 "stochastic.daf"
```

Under the Linux operating system INV\_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
inv_step1.exe "stochastic.daf"
```

For these commands, “INV\_STEP1” or “inv\_step1.exe” is the name of the executable program, and “stochastic.daf” is the name of the stochastic disposal action file.

### **24.2.2 STOCHASTIC Execution**

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the INV\_STEP1 code and writes a file named “Step1.Val” for use in the INV\_STEP2 code.

Under the Windows operating system (Releases 2000 or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC "Step1.Key"
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe "Step1.Key"
```

For these commands, “STOCHASTIC” or “stochastic.exe” is the name of the executable program, “Step1.Key” is the name of the input keyword file, and “Step1.Val” is the name of the output file of median values.

### **24.2.3 INV\_STEP2 Execution**

The final step in the processing sequence is to run the INV\_STEP2 code. This code reads the stochastic disposal action file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values disposal action file.

Under the Windows operating system (Releases 2000 or XP), INV\_STEP2 executes in a DOS box. A run of INV\_STEP2 is initiated by entering the following command line:

```
INV_STEP2 "stochastic.daf" "median.daf"
```

Under the Linux operating system INV\_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
inv_step2.exe "stochastic.daf" "median.daf"
```

For these commands, “INV\_STEP2” or “inv\_step2.exe” is the name of the executable program, “stochastic.daf” is the name of the stochastic disposal action file, and “median.daf” is the name of the median-values disposal action file.



## 25.0 Inventory Database

### 25.1 Overview

The SAC inventory database contains information derived from Hanford records, simulation of past Hanford processes (SIM – Soil Inventory Model), and projection of future waste generation (HTWOS – Hanford Tank Waste Operations Simulator). The structure of the SAC inventory database and the site naming conventions were derived from WIDS (Waste Information Data System). The SAC inventory database uses a modified version of the WIDS Site table as the basis for organizing waste site information. The SAC inventory database is implemented with SQL Server and uses Microsoft Access as an interface.

#### 25.1.1 Location in the Processing Sequence

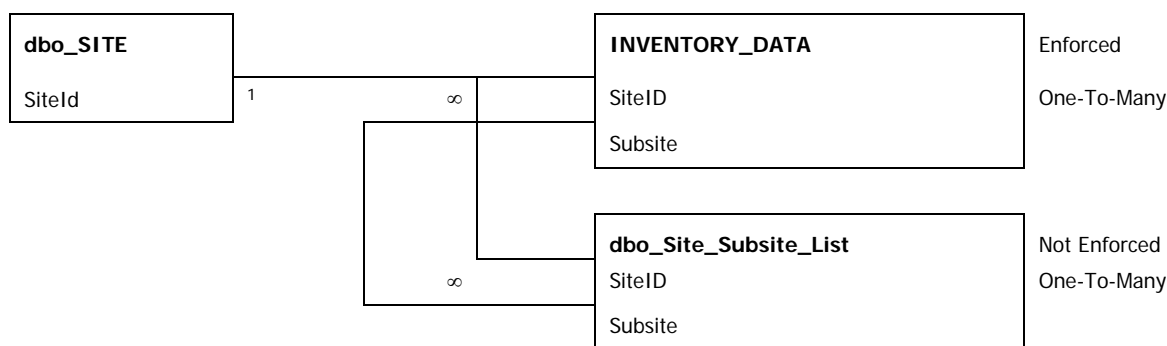
The SAC inventory database provides input for the INPROC utility code. It is the first processing step of any SAC simulation.

#### 25.1.2 Memory Requirements

The current size of the inventory database is 1.2 gigabytes. The memory required to run queries should not exceed 128 MB.

### 25.2 Database Structure

The inventory data is contained in three tables. These tables are Inventory\_Data, dbo\_SITE, and dbo\_Site\_Subsite\_List. The dbo\_SITE table is linked to the Inventory\_Data table by common SiteID. The dbo\_Site\_Subsite\_List is linked to the Inventory\_Data table through both the SiteID and Subsite fields. The relationships between these tables is shown in Figure 25.1.



**Figure 25.1** Relationships between Inventory Database Tables

The Inventory\_Data table is where the actual contaminant inventories and site volumes are stored. The fields in the Inventory\_Data table are listed in Table 25.1.

**Table 25.1** Field Descriptions for Table Inventory\_Data

Name	Type	Size	Description
Index	Long Integer	4	Autonumber field

Name	Type	Size	Description
SiteID	Long Integer	4	Site index from WIDS; additions numbered from 9900
SiteCode	Text	255	Descriptive Site name
Stat_Distribution	Text	50	Distribution parameter
Subsite	Text	50	Waste description or index
Attribute2	Text	50	Additional description; used only for waste transported offsite
Year	Long Integer	4	Year of contaminant deposit or basis
Constituent_name	Text	255	Radionuclide or chemical name
Constituent_value	Double	8	quantity
Constituent_units	Text	50	Ci, kg, or m <sup>3</sup> ; SIM Ci/m <sup>3</sup> or kg/m <sup>3</sup>
Source	Text	50	Origin of data (e.g., records, HTWOS, SIM)
Dataset	Text	50	Data package or file for traceability
Comment	Text	50	Comment – history of value
Action	Text	10	Field used to define data update action

The dbo\_SITE table is the list of all sites that are included in the SAC Inventory\_Data. This table provides information about the waste site where the inventory was generated. The fields in the table dbo\_SITE are given in Table 25.2.

**Table 25.2** Field Descriptions for Table dbo\_SITE

Field Name	Type	Size	Field Description
SiteId	Long Integer	4	Site index
SiteCode	Text	15	Unique code used to identify the site
SiteNames	Text	255	Common names by which site has been known
SiteType	Text	50	Type of structure for the site
SiteStatus	Text	20	Operational status of the unit
DesgArea	Text	20	Where the site is located at Hanford
OperUnit	Text	10	Operable unit in which site is located
SiteClassificationStatus	Text	20	Status within database: accepted, rejected, discovery
SitePackageStatus	Text	20	Data package validation
SiteReclassStatus	Text	25	Status of site after being changed from “accepted”
SiteDesc	Memo	-	Site Description
AssocStruct	Memo	-	Structures associated with site
SiteComment	Memo	-	Information not in any other fields
ReleaseDesc	Memo	-	Description of known releases from the site
ReleasePotDesc	Memo	-	Potential for unplanned release from site
EnvMonDesc	Memo	-	Environmental monitoring description
ProcessDesc	Memo	-	Process description
StartDate	Double	8	Facility start date
EndDate	Double	8	Facility end date
ValidatedBy	Text	40	Person performing technical review
ValidatedDate	Date/Time	8	Date technical review was completed
AssignedToFieldTech	Text	50	Person assigned to perform field investigation
DateSiteIdAssigned	Date/Time	8	Date that SiteID was assigned
InvestigationSubmissionDate	Date/Time	8	Date investigation was submitted
InvestigationAssignedDate	Date/Time	8	Date investigation was assigned
LocDesc	Memo	-	Location with relation to facilities, structures, roads,



Field Name	Type	Size	Field Description
			or other landmarks
PrioritizationLevel	Integer	2	Priority for investigation
SiteIDAssignedBy	Text	50	Person who assigned SiteID
SiteActivities	Memo	-	Remediation activities at the site
PreviousOperableUnit	Text	10	Previous operable unit designation
ReclassStatusDate	Date/Time	8	Date of the decision document
OperUnitPrintOrder	Double	8	Field used to control print order
SiteCodePrintOrder	Double	8	Field used to control print order
Site-Volume?	Yes/No	1	Does the site have a waste volume assigned?
Site-specific_Inventory?	Yes/No	1	Does the site have site-specific inventory?
Inv_consolidated-other_Site?	Yes/No	1	Has inventory been consolidated at another site?
Other_Site_containing_Inv	Text	50	Name of other site containing inventory
Screen_class-reclass	Yes/No	1	Does the site pass screening based on classification or reclassification?
Screen_SiteType	Yes/No	1	Is the site of a selected site type?
Associated_with_Duplicate?	Yes/No	1	Is the site associated with a duplicate?
Is_Duplicate?	Yes/No	1	Is this site the duplicate of another?
Duplicate_of_SiteCode	Text	50	Site Code of duplicate site
DISCARD_SITE	Yes/No	1	Should site inventory be dropped based on description?
DISCARD_JUSTIFICATION	Text	50	Reason for eliminating site inventory
No_Waste_Received?	Yes/No	1	Has this site received no waste?
Consolidated_Site?	Yes/No	1	Has the site been consolidated with other(s)?
Other_site_name	Text	50	Site of consolidation
Selected?	Yes/No	1	Is this site selected for inventory consideration?
Attributed_elsewhere?	Yes/No	1	Is the inventory for this site attributed elsewhere?
Selected_surrogate?	Yes/No	1	Has a surrogate been selected for site w/o inventory?
SITE_GROUP	Text	50	Origin of site/inventory; examples WIDS, HTWOS

The dbo\_Site\_Subsite\_List table is used to define the final disposal location for waste generated at a site. Not all waste forms have the same final disposal location, so this table contains a subsite field that identifies the specific waste type and its final disposal location. The fields in the dbo\_Site\_Subsite\_List are given in Table 25.3.

**Table 25.3** Field Descriptions for Table dbo\_Site-Subsite\_List

Name	Type	Size	Field Description
IDd	Decimal	16	Index
SiteID	Long Integer	4	Site index
SiteCode	Text	255	Unique code used to identify the site
Subsite	Text	50	Waste type (liquid, solid, etc.)
Final_Disposal_SiteID	Long Integer	8	SiteID of final waste disposal location
ERDF	Yes/No	1	Is waste from site disposed in ERDF?

## **25.3 Importing Data into the Inventory Database**

There are three major sources of values in the inventory database. Information concerning waste disposals or releases at Hanford from published documents, contractor records, or other databases are termed Record data. Simulations have been used to estimate disposal of some liquid wastes (the Soil Inventory Model, or SIM). Future waste from processing tank waste is projected by Hanford Tank Waste Operations Simulator (HTWOS).

### **25.3.1 Importing Record Data**

Record data are derived from various Hanford reports or databases and are collected into data packages. These data packages usually contain detailed instructions for adding new records and for deleting obsolete records when necessary. The data package may include a text document containing import instructions, a spreadsheet containing a list of values to be added or deleted, and may have a database to be incorporated into the Inventory\_Data table. These data are imported without the assistance of a utility code.

### **25.3.2 Importing SIM Files**

Data from the Soil Inventory Model (SIM) (Simpson et al. 2001) form the basis for the inventory for many sites that incurred liquid releases. A pre-processing module SIMS (see Section 0) reformats the SIM data into a suite of comma-separated variable files ready for import into the database. The files contain a stochastic representation for volume and concentrations (with 23 values from a user-defined statistical distribution) for each analyte.

### **25.3.3 Importing HTWOS Files**

Data from Hanford Tank Waste Operations Simulator (HTWOS) are imported as two comma-separated value files named "HTWOS\_TDP.CSV" and "HTWOS\_LEAK.CSV". These files contain HWTOS data already processed by HTWOS\_TLDP (Section 19.0) and HTWOS\_TDP (Section 18.0).

## **25.4 Exporting Records from the Inventory Database**

INPROC (Section 23.0) is the code that processes inventory information that is stored in the inventory database. Separate export procedures are required for waste or materials that 1) will be stored or disposed offsite, 2) have been disposed at Hanford, or 3) will be generated as separate waste streams but will be disposed at a common site. Data files exported from these three procedures must be combined into a single text file for use in the INPROC code. A form has been created to facilitate creation and export of a file named "inproc\_input". This comma-separated file is created in the same location as the database. The form that should be used for this task is called "INPROC\_INPUT\_CVS\_CREATE". Pressing the "Create file" button in this form executes the macro, "Create INPROC\_Input File". This macro executes a series of six queries. The queries used to create the INPROC input from the inventory data table are discussed in the following sections.

### **25.4.1 Query Create Inventory\_Data for Selected Analytes**

The first step in the creation of the INPROC input file is to select the desired data by setting selection criteria in this query. At this time, the disposal year is set to >1943 and the analyte list is set to C14,

Cl36, Cs137, Eu152, H3, I129, Np237, Pa231, Ra226, Se79, Sr90, Tc99, U233, U234, U235, U238, Pu\* (all plutonium isotopes), Am241, Am243. Volume records are also included in the selected data. If desired, this query could be used to set criteria for any field to limit the data to be used in a run of the SAC system.

#### **25.4.2 Query Create\_INPROC\_Onsite**

The onsite data is the inventory whose final disposal site is onsite. HTWOS liquid waste streams (HTWOS-CHE-LIQ, HTWOS-CHW-LIQ, HTWOS-LAW-LIQ, HTWOS-LERF-LIQ, HTWOS-NWTP-LIQ, HTWOS-RH-LIQ, and H2-ETF-LIQUID) are excluded from this selection but will be consolidated and included in the next step. The Inproc\_Onsite data also excludes inventory that is disposed of offsite (WIPP, YUCCA, OFFSITE\_TBD), special nuclear materials (SNM), waste that is sent to a solid waste facility (218-E-RCRA), and waste that is released to the atmosphere (291-WTP).

#### **25.4.3 Query Append HTWOS combined to INPROC\_onsite**

Waste streams (from waste treatment facilities) from HTWOS\_TDP that are disposed of onsite are sent to a solid waste facility (218-E-RCRA), a liquid waste facility (600-211), or are released to the atmosphere (291-WTP). The waste to be disposed of onsite must first be combined (by year and waste type) before it is removed to the combined disposal site. These database entries all have the text “HTWOS” in the source field.

#### **25.4.4 Query Append Stored Mixed Waste to INPROC\_onsite in site 218-E-RCRA**

Waste that is sent to the 218-E-RCRA solid waste facility is appended to the onsite inventory data in this step.

#### **25.4.5 Query Create INPROC\_Input**

The INPROC\_Input table is created in this step. The INPROC\_Input inventory consists of the INPROC\_Onsite inventory with deletions of sites that had only clean water discharges and sites with an associated volume but no inventory data. The 70 sites that have been labeled “clean” are typically described as sanitary sewage systems, septic systems, ponds, and well fields.

##### **25.4.5.1 Query 70\_Clean\_sites**

There are 70 sites included in the site list that had liquid releases but did not include any radioactivity. A list of the clean water discharge sites is compiled in this subquery so that these sites can be included with only a volume in the inventory. The list of the clean sites is stored in the “Clean\_Sites” spreadsheet in the Geographic and Operational Site Parameters List (GOSPL) for Hanford Assessments workbook.

##### **25.4.5.2 Query 161\_INPROC\_1-Constituent-subsite-year**

There are volume records included for some sites that do not have associated radioactive disposal. In this subquery, the list of sites that have a volume and no inventory—but are not on the “clean” site list—is compiled so that these sites can be excluded from the inventory.

### **25.4.5.3 Query Append waste to be transported offsite to INPROC\_Input**

This query identifies the waste that will be sent offsite and appends these records to the INPROC\_Input table. Only records whose final disposal site is WIPP, YUCCA, SNM, or OFFSITE\_TBD are included, and only those sites whose inventory includes radionuclide inventory and not just volume records.

### **25.4.5.4 Query Single Analyte Inventories**

There are volume records included for some sites that do not have associated radioactive disposal. In this subquery, the list of sites that have a volume and no inventory is compiled so that these sites can be excluded from the inventory.

## **26.0 INVEXT – Inventory Data Extractor**

### **26.1 Overview**

The INVEXT (Inventory Data Extractor) code allows the user to collect and summarize inventory information in an inventory “results” file output by the INVENTORY code. The most commonly used output is a table indicating inventory for each analyte subdivided by data type, where data type is record data, surrogate rule application, SIM data, or HTWOS data. This program also extracts data for the past leaks, future leaks, and residuals for each Hanford waste tank.

#### **26.1.1 Location in the Processing Sequence**

The INVEXT code reads data files written by the INVENTORY code.

#### **26.1.2 How the Code Is Invoked**

INVEXT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INVEXT executes in a DOS box. A run of INVEXT is initiated by entering the following command line:

```
INVEXT "Keyfilename"
```

Under the Linux operating system INVEXT is executed through any of the following Bourne Shell or C Shell commands:

```
invext.exe "Keyfilename"
```

For these commands, “INVEXT.EXE” or “invext.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVEXT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVEXT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **26.1.3 Memory Requirements**

The INVEXT code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVEXT code will require less than 8 MB of memory.

### **26.2 Data Files**

The INVEXT code reads four input files and writes up to three output files. These files are described in the following sections.

## 26.2.1 Input Files

The four input files for the INVEXT code are the following:

- **INVEXT Keyword File:** The INVEXT keyword file provides basic control information. The user may select which analytes to process, which sites to examine, and whether to also extract tank-specific information.
- **ESD Keyword File:** The ESD keyword file contains the control information that the inventory and environmental transport codes use to generate concentration data files.
- **Res File:** This file contains the inventory information for a single realization produced by the INVENTORY code for use in the SAC release and transport modules. The file format is described in Volume 1 of this document and is not described further here.
- **Rule File:** This data source rule file is produced by the INPROC code and identifies the type of rule that produced each waste data value.

### 26.2.1.1 InvExt Keyword File

The INVEXT keyword definition file contains control information for the desired calculation. An example file is provided in Table 26.1.

**Table 26.1** Example Keyword File for the INVEXT Code

```
REPORT " InvExt.Rpt"
TITLE "Extract inventory sums for a set of sites from a *.RES file"
USER "Carmen Arimescu"
!
! All data decay corrected to the following year (no chain decay)
DECAY 2070
!
VERBOSE
!DEBUG
!
! File definitions
FILE RES      "inv1.res"      ! Input   : The *.res file
FILE ESD      "ESD_CA_Best.key" ! Input   : The ESD keyword file
FILE RULE     "CA_Best.rul"   ! Input   : The data source rule file
FILE OUTPUT   "Test_Out.csv"  ! Output  : The summed inventory values
FILE OUTPUT_T "Tanks_Out.csv" ! Output  : The summed tank values
!
TIME START=1944 STOP=2070
!TIME ALL
!
! Analytes to process
ANALYTE ID="H3"
ANALYTE ID="C14"
ANALYTE ID="C136"
ANALYTE ID="Se79"
ANALYTE ID="Sr90"
ANALYTE ID="Tc99"
```

```
ANALYTE ID="I129"
ANALYTE ID="Cs137"
ANALYTE ID="Eu152"
ANALYTE ID="U238"
!
! Wastes to include in the sum
!WASTES ALL
WASTES LIST "liquid"
!
! Sites to include in the sum
SITES ALL
!SITES LIST "WIPP_218-E-12B" "WIPP_218-W-3AE" "200-E-28" "200-E-30"
!
LEAK YEAR=2004 RESIDUAL="cement"
!
TANK ID="4843"
TANK ID="100-B-3"
TANK ID="100-B-5"
TANK ID="100-B-8"
TANK ID="100-B-15"
TANK ID="100-C-3"
TANK ID="100-C-6"
TANK ID="100-D-3"
TANK ID="100-D-23"

! End of the keywords
END
```

### 26.2.1.2 ESD Keyword File

The INVEXT code also reads keywords from the environmental settings file. These keywords are read from a different file—and can have a different definition—than a keyword defined for internal use in the INVEXT code. The following ESD keywords are required by the INVEXT code:

- ANALYTE – definition of analytes in the environmental simulations
- END – end of the environmental settings keywords
- REALIZAT – number of realizations that were simulated
- TITLE – environmental simulation title.

### 26.2.1.3 Rule File

The rule file written by the INPROC code is a text file that identifies the data source as being an estimation rule, an input record, or a surrogate rule (the last entry on each line in the file). Subclasses of input records are SIM and HTWOS data. These subsets are identified using the first entry on each line. A subset of the records in this file for five disposal actions is provided in Table 26.2

**Table 26.2** Subset of the Records in the INPROC-Generated Rule File

"Records",	"100-N-60",	"liquid",	1987,	volume,	2.83870E-01,	sur
"Records",	"100-N-60",	"liquid",	1987,	H3	4.17216E-03,	sur
"Records",	"100-N-60",	"liquid",	1987,	C14	8.62755E-04,	sur
"Records",	"100-N-60",	"liquid",	1987,	Sr90	1.36145E-04,	sur
"Records",	"100-N-60",	"liquid",	1987,	Cs137	9.49126E-04,	sur
"Records",	"100-N-60",	"liquid",	1987,	Eu152	6.03644E-05,	sur
"Records",	"100-N-60",	"liquid",	1987,	U235	4.76000E-08,	sur
"Records",	"100-N-60",	"liquid",	1987,	U238	2.23267E-06,	sur
"Records",	"116-B-4",	"liquid",	1961,	volume,	2.50000E+01,	inp
"Records",	"116-B-4",	"liquid",	1961,	H3	4.17216E-03,	inp
"Records",	"116-B-4",	"liquid",	1961,	C14	8.62755E-04,	est
"Records",	"116-B-4",	"liquid",	1961,	Sr90	1.36145E-04,	inp
"Records",	"116-B-4",	"liquid",	1961,	Cs137	9.49126E-04,	inp
"Records",	"116-B-4",	"liquid",	1961,	Eu152	6.03644E-05,	est
"Records",	"116-B-4",	"liquid",	1961,	U235	4.76000E-08,	est
"Records",	"116-B-4",	"liquid",	1961,	U238	2.23267E-06,	inp
"Records",	"116-B-4",	"liquid",	1962,	volume,	2.50000E+01,	inp
"Records",	"116-B-4",	"liquid",	1962,	H3	4.17216E-03,	inp
"Records",	"116-B-4",	"liquid",	1962,	C14	8.62755E-04,	est
"Records",	"116-B-4",	"liquid",	1962,	Sr90	1.36145E-04,	inp
"Records",	"116-B-4",	"liquid",	1962,	Cs137	9.49126E-04,	inp
"Records",	"116-B-4",	"liquid",	1962,	U235	4.76000E-08,	est
"Records",	"116-B-4",	"liquid",	1962,	U238	2.23267E-06,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	volume,	4.38000E+00,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	H3	9.03160E-07,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	C14	8.02330E-10,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	Se79	3.85230E-11,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	Tc99	1.70030E-08,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	Sr90	1.72090E-05,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	I129	1.70000E-11,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	Pa231	1.62270E-12,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	U235	2.52000E-11,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	U238	1.14097E-09,	inp
"SIM8a",	"200-E-100",	"liquid",	1945,	Np237	1.05000E-10,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	volume,	1.05000E+02,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	C14	1.86667E-03,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Se79	2.30476E-04,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Tc99	1.07619E-03,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Sr90	3.40952E-03,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	I129	4.25714E-06,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Cs137	1.20952E-03,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Eu152	5.00000E-04,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Pa231	3.00952E-06,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	U235	2.02857E-06,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	U238	9.40952E-05,	inp
"HTWOS",	"218-E-RCRA",	"cement",	2008,	Np237	1.47619E-06,	inp

### 26.2.2 Output Files

The output files written by the INVEXT code are a report file and up to two results files.



### 26.2.2.1 INVEXT Report File

The report file contains summary information from the run of the INVEXT code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the INVEXT code.

### 26.2.2.2 INVEXT Site Results File

The INVEXT site results file contains the summed inventory for every selected analyte by inventory data rule for the set of selected sites. The results are all decay corrected to a common year. The file is written in comma-separated format for ease of importing into a spreadsheet program. An example file is provided in Table 26.3.

**Table 26.3** Example Site Results File from the INVEXT Code

Analyte	SIM Data	HTWOS Data	Record Data	Estimated Data	Surrogate Data	Total
C14	1.954970E+02	8.012610E+02	5.524116E+04	1.145416E+02	4.095726E+01	5.639341E+04
Cs137	4.425221E+04	9.213586E+06	1.370045E+07	3.847766E+04	2.548278E+03	2.299932E+07
Cl36	0.000000E+00	0.000000E+00	3.483770E+02	0.000000E+00	0.000000E+00	3.483770E+02
I129	4.724469E+00	4.279041E+01	1.730347E+01	6.046459E-01	8.776995E-03	6.543177E+01
Np237	5.378675E+01	1.273766E+02	9.244523E+01	0.000000E+00	8.198117E-03	2.736168E+02
Se79	2.147487E+00	9.226255E+01	8.982004E+01	6.455941E-01	7.782592E-03	1.848835E+02
Sr90	9.249055E+03	9.010369E+06	6.600646E+06	1.841693E+04	1.894604E+03	1.564058E+07
Tc99	6.867034E+02	2.675301E+04	3.881417E+03	2.375322E+01	3.916408E+00	3.134880E+04
H3	3.437134E+03	1.020090E+02	2.379466E+04	0.000000E+00	2.719746E+01	2.736100E+04
Eu152	1.275216E-01	5.935924E+01	6.566693E+01	1.531145E+00	5.670035E+00	1.323549E+02
U233	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
U235	3.093724E+00	9.321692E+00	9.626241E+01	6.681802E-01	2.602662E-01	1.096063E+02
U238	1.422503E+02	4.337508E+02	5.047515E+03	1.280927E-05	1.227393E+01	5.635790E+03

### 26.2.2.3 INVEXT Tank Results File

The optional INVEXT tank results file contains the inventory for every selected analyte by tank, divided into past leaks, future leaks, and residual waste categories. The results are all decay corrected to a common year. The file is written in comma-separated format for ease of importing into a spreadsheet program. Excerpted data for two tanks are provided in Table 26.4.

**Table 26.4** Excerpts from a Tank Results File from the INVEXT Code

TANK_ID	Analyte	Past Leak	Cleanout	Residual	Total
241-A-101	Volume	0.000000E+00	0.000000E+00	1.020000E+01	
241-A-101	C14	0.000000E+00	0.000000E+00	8.229977E-03	8.229977E-03
241-A-101	Cs137	0.000000E+00	0.000000E+00	1.511279E+02	1.511279E+02
241-A-101	Cl36	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
241-A-101	I129	0.000000E+00	0.000000E+00	2.929995E-04	2.929995E-04
241-A-101	Np237	0.000000E+00	0.000000E+00	1.499981E-01	1.499981E-01
241-A-101	Se79	0.000000E+00	0.000000E+00	1.989915E-03	1.989915E-03
241-A-101	Sr90	0.000000E+00	0.000000E+00	5.151646E+03	5.151646E+03
241-A-101	Tc99	0.000000E+00	0.000000E+00	2.879625E-01	2.879625E-01
241-A-101	H3	0.000000E+00	0.000000E+00	2.602117E-03	2.602117E-03
241-A-101	Eu152	0.000000E+00	0.000000E+00	1.187445E-01	1.187445E-01
241-A-101	U233	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

241-A-101,U235	,	0.000000E+00	,	0.000000E+00	,	2.760000E-03	,	2.760000E-03
241-A-101,U238	,	0.000000E+00	,	0.000000E+00	,	1.293000E-01	,	1.293000E-01
241-A-102,Volume	,	0.000000E+00	,	0.000000E+00	,	1.020000E+01		
241-A-102,C14	,	0.000000E+00	,	0.000000E+00	,	4.358802E-03	,	4.358802E-03
241-A-102,Cs137	,	0.000000E+00	,	0.000000E+00	,	7.039378E+02	,	7.039378E+02
241-A-102,C136	,	0.000000E+00	,	0.000000E+00	,	0.000000E+00	,	0.000000E+00
241-A-102,I129	,	0.000000E+00	,	0.000000E+00	,	3.699993E-04	,	3.699993E-04
241-A-102,Np237	,	0.000000E+00	,	0.000000E+00	,	2.739965E-02	,	2.739965E-02
241-A-102,Se79	,	0.000000E+00	,	0.000000E+00	,	2.179917E-03	,	2.179917E-03
241-A-102,Sr90	,	0.000000E+00	,	0.000000E+00	,	5.495089E+03	,	5.495089E+03
241-A-102,Tc99	,	0.000000E+00	,	0.000000E+00	,	3.389559E+00	,	3.389559E+00
241-A-102,H3	,	0.000000E+00	,	0.000000E+00	,	3.897909E-03	,	3.897909E-03
241-A-102,Eu152	,	0.000000E+00	,	0.000000E+00	,	5.653271E-03	,	5.653271E-03
241-A-102,U233	,	0.000000E+00	,	0.000000E+00	,	0.000000E+00	,	0.000000E+00
241-A-102,U235	,	0.000000E+00	,	0.000000E+00	,	1.790000E-02	,	1.790000E-02
241-A-102,U238	,	0.000000E+00	,	0.000000E+00	,	8.390000E-01	,	8.390000E-01

## 26.3 Keyword Definitions for INVEXT

In general, the keywords for the INVEXT code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 26.3.1 ANALYTE Keyword for INVEXT

The ANALYTE keyword is used to define the analytes for which inventory data will be extracted. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"]
```

Analytes chosen must be a subset of the analytes used in the fate and transport activities controlled through the ESD keyword file. Multiple analyte keywords may be entered. A separate ANALYTE keyword is required for each desired analyte. The following suite of ANALYTE keywords define three analytes for which inventory information is to be extracted:

```
ANALYTE ID="Tc99"
ANALYTE ID="Sr90"
ANALYTE ID="U238 "
```

### 26.3.2 END Keyword for INVEXT

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 26.3.3 FILE Keyword for INVEXT

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"} ... {modifier6="quote 6"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code. The modifiers associated with the FILE keyword are given in Table 26.5. The file name associated with a modifier must be entered before the next modifier is entered.

**Table 26.5** Modifiers Associated with the FILE Keyword in the INVEXT Keyword File

Modifier	Description
ESD	The quote string associated with the ESD modifier defines the location of the input ESD keyword file. Analyte half-life data are taken from this file.
RULE	The quote string associated with the RULE modifier defines the location of the input source data rule file produced by the INPROC code.
RES	The quote string associated with the RES modifier defines the location of the input results data file produced by the INVENTORY code.
OUTPUT	The quote string associated with the OUTPUT modifier defines the name of the output data file that contains the matrix of summed inventory values by analyte.
OUTPUT_T	The quote string associated with the OUTPUT_T modifier defines the name of the optional output data file that contains the matrix of inventory values by tank site.
OUTPUT_V	The quote string associated with the OUTPUT_V modifier defines the name of the optional output data file that contains the volume data by site and analyte.

### 26.3.4 LEAK Keyword for INVEXT

The optional LEAK keyword is required when one or more tanks are defined using the TANK keyword. The following is this keyword's syntax:

```
LEAK [YEAR= "T"] [RESIDUAL="quote"]
```

The calendar year associated with the YEAR modifier defines the year for which to determine whether a liquid release at a tank is a past leak or a future leak. A value of 2005 makes a release in all years before 2005 a past leak, and releases in the year 2005 and future years will be future leaks. The quote string associated with the RESIDUAL modifier identifies the waste form model for tank residuals. Currently a value of "cement" is appropriate. An example use of this keyword is the following:

```
LEAK YEAR=2004 RESIDUAL="cement"
```

### 26.3.5 REPORT Keyword for INVEXT

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 26.3.6 TANK Keyword for INVEXT

The optional TANK keyword is used to define the tanks for which data will be extracted. The following is this keyword's syntax:

```
TANK [ID="quote"]
```

The single quote string is associated with the ID modifier and must be an ID from the set of the tanks in the inventory source file. Multiple TANK keywords are required to define multiple tanks. Examples of this keyword are the following:

```
TANK ID="241-AX-102"  
TANK ID="241-AX-103"
```

### 26.3.7 TIME Keyword for INVEXT

The TIME keyword identifies the times (years) at which the calculations are to be performed for a single case. The following is this keyword's syntax:

```
TIME START=[T] STOP=[T]  
TIME ALL
```

All times in the inventory source file (\*.res) are used when the modifier ALL is present. A range of specific years can be defined by entering the modifiers START and STOP to identify the years of interest. Data for all years are included in the first example shown below. The second example extracts only data in the range of years from 1944 through 1960.

```
TIME ALL  
TIME START=1944 STOP=1960
```

### 26.3.8 WASTES Keyword for INVEXT

The WASTES keyword identifies the waste types for which data are to be extracted and summed. The following is this keyword's syntax:

```
WASTES [ALL | LIST ["quote1"] {"quote2"} ... {"quoten"}]
```

All waste types in the inventory results file are used when the modifier ALL is present. A list of specific waste types can be defined by entering the modifier LIST and a set of alphanumeric values identifying the waste types of interest. The waste type names are not case sensitive. Two examples of this keyword are the following:

```
WASTES ALL  
WASTES LIST "liquid" "cement"
```

### 26.3.9 SITES Keyword for INVEXT

The SITES keyword identifies the site names for which the calculations are to be performed. The following is this keyword's syntax:

```
SITES [ALL | LIST ["quote1"] {"quote2"} ... {"quoten"}]
```

All sites in the inventory source file are used when the modifier ALL is present. A list of specific sites can be defined by entering the modifier LIST and a set of alphanumeric values identifying the sites of interest. Sites are ignored if they are entered in this list, but no data are defined in the inventory results file. Two examples of this keyword are the following:

```
SITES ALL  
SITES LIST "200-E-28" "200-E-30"
```

### 26.3.10 TITLE Keyword for INVEXT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the INVEXT code."
```

### 26.3.11 USER Keyword for INVEXT

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 26.3.12 VERBOSE Keyword for INVEXT

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. The following is this keyword's syntax:

```
VERBOSE
```

### **26.3.13 VOLUME Keyword for INVEXT**

The presence of the optional VOLUME keyword initiates additional output of volume data to a file. The file name is specified using the OUTPUT\_V modifier on the FILE keyword. The following is this keyword's syntax:

VOLUME

## **27.0 INVPLLOT – Inventory Plotting**

### **27.1 Overview**

This program generates files of inventory information on an annual time step for combinations of analytes, waste sites, and waste locations. The generated files can be used in a plotting program. The INVPLLOT program reads the accumulated, decay-corrected files generated by INGRAB data extractor.

#### **27.1.1 Location in the Processing Sequence**

The INVPLLOT code reads data files written by the INVENTORY and INGRAB codes, and thus cannot be executed until these two codes have completed.

#### **27.1.2 How the Code Is Invoked**

INVPLLOT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INVPLLOT executes in a DOS box. A run of INVPLLOT is initiated by entering the following command line:

```
INVPLLOT "Keyfilename"
```

Under the Linux operating system INVPLLOT is executed through any of the following Bourne Shell or C Shell commands:

```
invplot.exe "Keyfilename"
```

For these commands, “INVPLLOT.EXE” or “invplot.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVPLLOT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVPLLOT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **27.1.3 Memory Requirements**

The INVPLLOT code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVPLLOT code will require less than 5 MB of memory.

### **27.2 File Definitions**

The INVPLLOT program reads three or more input files and writes three or more output files. These files are described in the following sections.

## 27.2.1 Input Files

The INV PLOT program reads a control keyword file and the “inventory.all” file written by the INVENTORY code. In addition, it reads one or more files written by the INGRAB code. The INV PLOT keyword file contains control information. An example keyword file that extracts data for two cases is provided in Table 27.1. Detailed definitions of the keywords are provided in Section 27.3.

**Table 27.1** Example Keyword File for INV PLOT

```
! Keyword file for INV PLOT
REPORT "InvPlot_U238.Rpt"
TITLE "Inventory Plots for SAC. Rev. 1 History 5 Inventory"
USER "Paul W. Eslinger"
! Index file output by INGRAB
FILE INDEX "Inventory.All"
! File for uncertainty data
FILE UNCERT "U238_Uncert.Rpt"
! Type of output
METRIC REALIZATION 1
! Maximum number of curves in any case
!MAXCURVE = 20
! Additional stuff written to the report file
VERBOSE
! Set of curves for all plots
CURVE LABEL="Liquid"      WASTE="liquid"
CURVE LABEL="Glass"      WASTE="glass"
CURVE LABEL="Other"      WASTE="capsul" "store" "TRUSLG"
CURVE LABEL="Reactor"    WASTE="Core" "rxcomp"
CURVE LABEL="Residual"   WASTE="cake"
CURVE LABEL="Air"        WASTE="gas"
CURVE LABEL="Offsite"    WASTE=" HLW" "NM1" "NM2" "NM4" "NM5" "NM6"
                        "NMd" "NMl" "NMs" "SF" "TRUf" "TRUn"
                        "TRUp" "TRUsf" "WIPP" "offsite"
CURVE LABEL="River"      WASTE="River"
CURVE LABEL="Soil"       WASTE=" soil" "soillf" "soilln" "soil3f"
                        "soil3n" "soils" "soilsf" "soilsn"
                        "soilss" "soilst" "soiltu"
CURVE LABEL="Cement"     WASTE=" cement" "cmntcs" "cmntct" "cmntcu"
                        "cmntg3" "cmntH2"
CURVE LABEL="All Types" WASTE="All_Types"
! End of the initial stuff
ENDINIT

CASE
TITLE="U238 Inventory - 300 Site"
ANALYTE "U238"
FILE PLOT "U238_300.csv"
FILE DATA "INGRABA_U238_ALL.CSV"
SITES
"300_RLWS" "300_RRLWS" "300_VTS" "300-121" "300-123"
"300-16" "300-2" "300-214" "300-224" "300-24"
```



```
"300-249" "300-25" "300-251" "300-255" "300-262"
"300-264" "300-265" "300-270" "300-28" "300-33"
"300-39" "300-4" "300-40" "300-48" "300-80"
"303-K_CWS" "303-M-SA" "303-M-UOF" "305-B_SF" "307_RB"
"309-WS-1" "309-WS-2" "313_ESSP" "316-1" "316-2" "316-3"
"316-5" "325_WTF" "331_LSLDF" "331_LSLT2"
"333_ESHWSA" "3712_USSA" "600-117" "UPR-300-1"
"UPR-300-10" "UPR-300-11" "UPR-300-12" "UPR-300-2"
"UPR-300-32" "UPR-300-34" "UPR-300-36" "UPR-300-37"
"UPR-300-38" "UPR-300-39" "UPR-300-4" "UPR-300-40"
"UPR-300-45" "UPR-300-48" "UPR-300-5" "UPR-300-FF-1"
ENDCASE

CASE
TITLE="U238 Inventory - Entire Site"
ANALYTE "U238"
FILE PLOT "All_U238_300.csv"
FILE DATA "N:\CA1_median\inventory\ingrab\IngrabA_U238.csv"
SITES ALL
ENDCASE

END
```

## 27.2.2 Output Files

The INVPLLOT program writes three or more output files. One file is a report file that contains text information about the run of the code. It is not described further here. Another output file is the uncertainty file that contains a summary of all cases performed during the code run. An additional output file is written for every CASE definition. This output file contains the inventory information formatted in comma-separated format for ease of input into a plotting program. The file starts with nine lines of identification information on file names and the code run. This identification information is followed by a data matrix, one line per year, of the inventory by waste form for each waste form curve.

## 27.3 Keyword Definitions for the INVPLLOT Code

The following restrictions apply to keyword order for the INVPLLOT code. The keywords are divided into two sections. The first section, called the control section, is entered once for every run of the code:

- The control section starts with the REPORT keyword and is terminated by an ENDINIT keyword. The other keywords in the control section can be entered in any order.
- The inventory plots are divided into different cases. Each plot definition starts with a CASE keyword and ends with an ENDCASE keyword. All other keywords for a case can be entered in any order.
- The END keyword must be the last keyword in the file.

## 27.3.1 Control Section Keywords for the INVPLLOT Code

### 27.3.1.1 CURVE Keyword for INVPLLOT

The CURVE keyword identifies a set of plot curves that apply to all cases. The following is this keyword's syntax:

```
CURVE [LABEL="quote 1"] [WASTE= "quote 2" ... "quote n"]
```

The plot curve label is entered in the quote string associated with the modifier LABEL. This label is for descriptive purposes and typically is used in the legend of a plot. The waste types to combine for this curve are defined by entering the modifier WASTE followed by a series of waste type IDs. The data for all the waste types will be summed before the plot curve is written. The single waste type ID of "All\_Types" can be used for a sum across all waste types. Waste type identifiers are case sensitive and must match with waste types contained in the inventory index file.

The following set of keyword entries define 11 curves to be used for all plots:

```
CURVE LABEL="Liquid"      WASTE="liquid"
CURVE LABEL="Glass"      WASTE="glass"
CURVE LABEL="Other"      WASTE="capsul" "store" "TRUSLG"
CURVE LABEL="Reactor"    WASTE="Core" "rxcomp"
CURVE LABEL="Residual"   WASTE="cake"
CURVE LABEL="Air"        WASTE="gas"
CURVE LABEL="Offsite"    WASTE= "HLW" "NM1" "NM2" "NM4" "NM5" "NM6" "NMd"
"NM1" "NM5" "SF" "TRUf" "TRUn" "TRUp" "TRUsf" "WIPP" !"offsite"
CURVE LABEL="River"      WASTE="River"
CURVE LABEL="Soil"       WASTE= "soil" "soil1f" "soil1n" "soil3f" "soil3n"
"soils" "soilsf" "soilsn" "soilss" "soilst" "soiltu"
CURVE LABEL="Cement"     WASTE= "cement" "cmntcs" "cmntct" "cmntcu"
"cmntg3" "cmnth2"
CURVE LABEL="All Types"  WASTE="All_Types"
```

### 27.3.1.2 ENDINIT Keyword for INVPLLOT

The ENDINIT keyword signifies the end of the control section keyword data. The following is this keyword's syntax:

```
ENDINIT
```

### 27.3.1.3 FILE Keyword for INVPLLOT

The FILE keyword is used to enter the name of the inventory index file. This file is written by the inventory code and always has the name "inventory.all". The following is this keyword's syntax:

```
FILE [INDEX="quotel"]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The inventory index file name is associated with the modifier INDEX. An example use of this keyword is the following:

```
FILE INDEX "N:\CA1_median\inventory\inventory.all"
```

#### 27.3.1.4 MAXCURVE Keyword for INVLOT

The INVLOT code defaults to allowing 20 inventory curves per case. The optional MAXCURVE keyword can be used to increase the number of curves allowed in each case. The following is this keyword's syntax:

```
MAXCURVE [N1]
```

The single integer entered on this keyword identifies the maximum number of curves to allow for all cases. The following example allows 23 curves to be defined for each case.

```
MAXCURVE 23
```

#### 27.3.1.5 METRIC Keyword for INVLOT

The METRIC keyword is used to define the output metric for the inventory data. The following is this keyword's syntax:

```
METRIC [ MEAN | REALIZAT=N1 | HORSETAIL ]
```

If the single modifier MEAN is used, then the average (mean) of all data realizations is output. If the single modifier REALIZAT is used, followed by an integer, then the data for that specific realization will be output. If the modifier HORSETAIL is used, the data for all realizations will be output. The following examples illustrate the use of all definitions for this keyword:

```
METRIC MEAN  
METRIC REALIZAT = 3  
METRIC HORSETAIL
```

#### 27.3.1.6 REPORT Keyword for INVLOT

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

#### 27.3.1.7 TITLE Keyword for INVLOT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the INVPLLOT code."
```

#### **27.3.1.8 USER Keyword for INVPLLOT**

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

#### **27.3.1.9 VERBOSE Keyword for INVPLLOT**

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

### **27.3.2 CASE Section Keywords for the INVPLLOT Code**

The plots of inventory are divided into different cases. Each plot definition starts with a CASE keyword and ends with an ENDCASE keyword. All other keywords for a case can be entered in any order. The keywords defined for a case are described in the following sections.

#### **27.3.2.1 ANALYTE Keyword for INVPLLOT**

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ "quote 1" ]
```

The single quote string must be an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. An example of this keyword is the following:

```
ANALYTE "H3"
```

#### **27.3.2.2 CASE Keyword for INVPLLOT**

The CASE keyword is used to signify the start of a plot case definition. The following is this keyword's syntax:

```
CASE
```

### 27.3.2.3 ENDCASE Keyword for INVPLLOT

The ENDCASE keyword is used to signify the end of a plot case definition. The following is this keyword's syntax:

```
ENDCASE
```

### 27.3.2.4 FILE Keyword for INVPLLOT

The FILE keyword is used to enter the name of the INGRAB-produced inventory file and the name of the output file. The following is this keyword's syntax:

```
FILE {PLOT="quote 1"} {DATA="quote 2"}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The inventory data file name is associated with the modifier DATA. Typically the data file is an accumulated, decay-corrected output from the INGRAB program. The output data file name is associated with the modifier PLOT. A file name ending in “.csv” is recommended. Both files can be defined using one FILE keyword, or they can be entered using separate keywords. An example use of this keyword is the following:

```
FILE PLOT "All_U238_300.csv"
```

```
FILE DATA "N:\CA1_median\inventory\ingrab\IngrabA_U238.csv"
```

### 27.3.2.5 SITES Keyword for INVPLLOT

The SITES keyword identifies a set of sites to include in the data sum for a plot. The following is this keyword's syntax:

```
SITES [ ALL | "quote 1" {"quote 2" ... "quote n"} ]
```

One or more sites are identified using a single SITES keyword. Each site is identified by entering the site ID enclosed in a set of double quotation marks. All sites can be identified by entering the modifier ALL. The following example identifies 19 sites to include in the data analysis:

```
SITES "300_RLWS" "300_RRLWS" "300_VTS" "300-121"  
      "300-123" "300-16" "300-2" "300-214" "300-224"  
      "300-24" "300-249" "300-25" "300-251" "300-255"  
      "300-262" "300-264" "300-265" "300-270" "300-28"
```

### 27.3.2.6 TITLE Keyword for INVPLLOT

The TITLE keyword is used to define a single-line title used for labeling the output for this case. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. This title is different for each plot and is different than the title in the control keyword section. The following example defines a title for a particular case:

```
TITLE "U238 - Entire Site"
```

### **27.3.3 END Keyword for INV PLOT**

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

## **28.0 INVREPAR – Inventory Data Repartitioning**

### **28.1 Overview**

The INVREPAR (Inventory Data Repartitioning) code allows the user to split the inventory at one site into multiple sites. The nominal use of this code is to split the inventory for B Pond liquid wastes into a main disposal and three additional lobes for a user-specified range of years.

#### **28.1.1 Location in the Processing Sequence**

The INVREPAR code reads inventory data files (\*.res files) written by the INVENTORY code.

#### **28.1.2 How the Code Is Invoked**

INVREPAR can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INVREPAR executes in a DOS box. A run of INVREPAR is initiated by entering the following command line:

```
INVREPAR "Keyfilename"
```

Under the Linux operating system INVREPAR is executed through any of the following Bourne Shell or C Shell commands:

```
invrepar-1.exe "Keyfilename"
```

For these commands, “INVREPAR.EXE” or “invrepar-1.exe” is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVREPAR is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVREPAR cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

The INVREPAR code must be run in the directory that contains the inventory results files if the default file naming convention is used. Otherwise, all file names can contain path information in the name.

#### **28.1.3 Memory Requirements**

The INVREPAR code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVREPAR code will require less than 2 MB of memory.

### **28.2 Data Files**

The INVREPAR code reads two or more input files and writes two or more output files. These files are described in the following sections.

### 28.2.1 Input Files

The input files for the INVREPAR code are the following:

- **INVREPAR Keyword File:** The INVREPAR keyword file provides basic control information. The user may select which site to partition and the destination sites.
- **Results File:** This file contains the inventory information for a single realization produced by the INVENTORY code for use in the SAC release and transport modules. The file format is described in Volume 1 of this document and is not described further here. More than one “\*.res” file can be processed in a single run of the code.
- **Rule File:** The inventory rule file generated by the INPROC code is used under some input data configurations.

The INVREPAR keyword definition file contains control information for the desired calculation. An example file is provided in Table 28.1.

**Table 28.1** Example Keyword File for the INVREPAR Code

```
FILE REPORT "InvRePar.rpt"
USER "Paul W. Eslinger"
FILE EXPLICIT "inv1.res"
VERBOSE
OS "Unix"
! Define the source site to partition
SOURCE ID="216-B-3" FROM 1980 TO 1990
! Define the destination sites to receive the repartitioned waste
DESTINATION ID="216-B-3" FRACTION=0.25
DESTINATION ID="216-B-3A_RAD" FRACTION=0.35
DESTINATION ID="216-B-3B_RAD" FRACTION=0.20
DESTINATION ID="216-B-3C_RAD" FRACTION=0.20
END
```

### 28.2.2 Output Files

The output files written by the INVREPAR code are a report file, one or more inventory results files, and an optional revised inventory rule file. The report file contains summary information from the run. This file contains a record of the code version, time and date of run execution, input and output file names, and run information. This file also contains any error messages generated by the INVREPAR code.

One or more revised inventory files are produced by this code. The file format for these files is described in Volume 1 of this document and is not described further here. The file name of each revised inventory file is unchanged from the original file.

The format of the output rule file identical to the input rule file. The file format for these files is described in Volume 1 of this document and is not described further here.



## 28.3 Keyword Definitions for INVREPAR

In general, the keywords for the INVREPAR code can be entered in any order. The only restriction is that the END keyword terminates keyword entry; therefore it should be the last entry in the file.

### 28.3.1 DESTINAT Keyword for INVREPAR

The DESTINAT(ION) keyword identifies the sites to which portions of the source site will be routed. The following is this keyword's syntax:

```
DESTINAT [ID="quote1"] [FRACTION=N1]
```

The ID of the destination site is entered in the quote string associated with the ID modifier. The fraction of the waste from the source site that ends up at this destination site is given by the numerical value associated with the FRACTION modifier. Fractions must be in the range 0 to 1. The sum of fractions from all DESTINAT keywords must be one. Every destination site is defined using a separate DESTINAT keyword. Four examples of this keyword are the following:

```
DESTINATION ID="216-B-3" FRACTION=0.25  
DESTINATION ID="216-B-3A_RAD" FRACTION=0.35  
DESTINATION ID="216-B-3B_RAD" FRACTION=0.20  
DESTINATION ID="216-B-3C_RAD" FRACTION=0.20
```

All waste streams from the source site are routed to the destination sites. The source site must also be a destination site. If all of the waste is moved to other sites, the source site must be included as a destination site with a fraction of zero. This version of the code requires a destination site other than the source site to have an initial zero inventory.

### 28.3.2 END Keyword for INVREPAR

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 28.3.3 FILE Keyword for INVREPAR

The FILE keyword is used to define the names of input and output files. There are four forms for this keyword's syntax:

```
FILE [REPORT="quote 1"]  
FILE [EXPLICIT="quote 1"]  
FILE [REALIZAT] [FROM=N1] [TO=N2] [BASIS=N3]  
FILE [RULES] [INPUT="quote 1"] [OUTPUT="quote 2"]
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code. The modifiers associated with the FILE keyword are provided in Table 28.2.

**Table 28.2** Modifiers Associated with the FILE Keyword in the INVREPAR Keyword File

Modifier	Description
REPORT	The quote string associated with the REPORT modifier defines the name of the report (log) file. Only one file can be defined with an entry of this keyword when the REPORT modifier is used.
EXPLICIT	The quote string associated with the EXPLICIT modifier defines the name of the inventory results file that will be modified. This version of the file keyword should not be used when the REALIZAT modifier is used. Only one file can be defined with an entry of this keyword when the EXPLICIT modifier is used.
REALIZAT	The REALIZAT modifier instructs the code to develop default file names for one or more inventory results files. The numerical values associated with the FROM and TO modifiers identifies the range of realizations for which inventory data will be partitioned. The numerical value associated with the BASIS modifier identifies the largest realization number in the inventory run that generated the results files. This version of the file keyword should not be used when the EXPLICIT modifier is used.
RULES	The quote string associated with the INPUT modifier defines the name of the input inventory rule file containing data to be modified. The quote string associated with the OUTPUT modifier defines the name of the output inventory rule file containing the modified data.

The report file for a run of the code could be defined using the following keyword:

```
FILE REPORT "InvRePar.rpt"
```

The following keyword would cause the INVREPAR code to operate only on one explicitly named inventory results file.

```
FILE EXPLICIT "inv1.res"
```

The following keyword would cause the INVREPAR code to operate on the separate files containing inventory in realizations 1 through 100 that were generated by an inventory code run that processed exactly 100 realizations.

```
FILE REALIZAT FROM 1 TO 100 BASIS 100
```

The following optional keyword would cause the INVREPAR code to modify entries in the the file "Test\_rule.dat" and write the values to the file "Test\_rule\_revised.dat".

```
FILE RULES INPUT="Test_rule.dat" OUTPUT="Test_rule_revised.dat"
```

In practice, a run of the code for a 100-realization simulation case might have the following three FILE keywords.

```
FILE REPORT "InvRePar.rpt"  
FILE REALIZAT FROM 1 TO 100 BASIS 100  
FILE RULES INPUT="Test_rule.dat" OUTPUT="Test_rule_revised.dat"
```

### 28.3.4 OS Keyword for INVREPAR

The OS keyword is used to define the operating system under which the code is executed. The following is this keyword's syntax:

```
OS [ "Unix" | "Windows" ]
```

There is no default for the OS keyword. One of the operating systems must be chosen. The following keyword selects the Unix (Linux) operating system

```
OS "Unix"
```

### 28.3.5 SOURCE Keyword for INVREPAR

The SOURCE keyword identifies the source site that will be portioned to several sites. The following is this keyword's syntax:

```
SOURCE [ ID="quote1" ] [ FROM=N1 ] [ TO=N2 ]
```

The ID of the source site is entered in the quote string associated with the ID modifier. Partitioning can only be performed for a contiguous set of years. The first year is defined by the numerical value associated with the FROM modifier. The last year is defined with the numerical value associated with the TO modifier. An example of this keyword that partitions waste in the years 1980 through 1990 is the following:

```
SOURCE ID="216-B-3" FROM 1980 TO 1990
```

### 28.3.6 USER Keyword for INVREPAR

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 28.3.7 VERBOSE Keyword for INVREPAR

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. The following is this keyword's syntax:

```
VERBOSE
```



## **29.0 INVSUM – Inventory Data Summations**

### **29.1 Overview**

The INVSUM (Inventory Data Summation) data extractor allows the user to collect and summarize inventory information. The most commonly used output is a table indicating which sites have a nonzero inventory for each analyte. Sums of data by site and waste type can also be obtained.

#### **29.1.1 Location in the Processing Sequence**

The INVSUM code reads data files written by the INVENTORY and INGRAB codes, thus cannot be executed until these two codes have completed.

#### **29.1.2 How the Code Is Invoked**

INVSUM can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), INVSUM executes in a DOS box. A run of INVSUM is initiated by entering the following command line:

```
INVSUM "Keyfilename"
```

Under the Linux operating system INVSUM is executed through any of the following Bourne Shell or C Shell commands:

```
invsum.exe "Keyfilename"
```

For these commands, "INVSUM.EXE" or "invsum.exe" is the name of the executable program, and "Keyfilename" is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVSUM is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVSUM cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **29.1.3 Memory Requirements**

The INVSUM code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVSUM code will require less than 5 MB of memory.

### **29.2 File Definitions**

The INVSUM code reads three or more input files and writes two output files. The number of input files depends on the number of contaminants being analyzed. These files are described in the following sections.

## 29.2.1 Input Files

The input files for the INVSUM code are a keyword file, a inventory index file, and one or more inventory data files created by the INGRAB code.

### 29.2.1.1 INVSUM Keyword File

The INVSUM keyword definition file contains control information for the desired data extraction. Individual keywords are defined in Section 29.3. Table 29.1 contains an example keyword file for the INVSUM code.

**Table 29.1** Example Keyword File for INVSUM

```
! Keyword file for the InvSum program
REPORT "PWE-median.Rpt"
TITLE "Inventory Sums by Site for SAC. Rev. 1 Inventory"
USER "Paul W. Eslinger"
! Index file output by INGRAB
FILE INDEX "O:\PWE\inventory-median\Inventory.All"
FILE SUM "Prelim5-median.csv" !Output file
! Analyte specific data
ANALYTE ID="H3" FILE="O:\PWE\inventory-median\INGRABA_H3_ALL.CSV"
ANALYTE ID="Tc99" FILE="O:\PWE\inventory-median\INGRABA_Tc99_ALL.CSV"
ANALYTE ID="Sr90" FILE="O:\PWE\inventory-median\INGRABA_Sr90_ALL.CSV"
ANALYTE ID="U238" FILE="O:\PWE\inventory-median\INGRABA_U238_ALL.CSV"
ANALYTE ID="I129" FILE="O:\PWE\inventory-median\INGRABA_I129_ALL.CSV"
! Define the waste stream to be summed
WASTE "All_Types"
! Year for output data
YEAR 2050
! Additional stuff written to the report file
VERBOSE
! End of the keywords
END
```

### 29.2.1.2 Inventory Index File

The inventory index file used by INGRAB is written by the INVENTORY code. The nominal name of this file is "inventory.all". The following information is extracted from this file:

- The number of years of inventory data and the suite of calendar years
- The number of sites included in the analysis and a list of the site IDs
- The number of analytes in the analysis and a list of the analyte IDs
- The number of waste types in the analysis and a list of the waste type IDs.

### 29.2.1.3 Accumulated Inventory Files

A separate accumulated inventory file is required for each analyte in this analysis. These files are all written by the INGRAB code. These files contain inventory values, accumulated to represent the total released for a given site at a given year. The inventory data in these files are decay corrected for

radioactive analytes. The accumulated inventory is extracted from these files for the year specified in the INVSUM keyword file.

## 29.2.2 Output Files

The output files written by the INVSUM code are a report file, a results file, and an optional summary statistics file.

### 29.2.2.1 INVSUM Report File

The report file contains summary information from the run of the INVSUM code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the INVSUM code.

### 29.2.2.2 INVSUM Results File

The INVSUM results file contains the accumulated inventory for every waste site for the specified waste type and year. It contains data for every realization generated by the INVENTORY code. Table 29.2 contains excerpted records from an INVSUM results file for the single analyte tritium (tritium, H3) and a single realization. The file is written in comma-separated format for ease of importing into a spreadsheet program.

**Table 29.2** Excerpted Records from an INVSUM Results File

```
"Program Name:", "InvSum"  
"Program Version:", "1.00.A.1"  
"Program Date:", " 6 Aug 2003"  
"Run ID:", "20030925111335"  
"Keyword File Name:", "PWE-median.key"  
"Analyte:", "H3"  
"Inventory File:", "O:\PWE\inventory-median\INGRABA_H3_ALL.CSV"  
"Waste Form:", "All Types"  
"atmosphere", 0.000E+00  
"store", 0.000E+00  
"216-U-1%2-Fast", 0.000E+00  
"100-B-15", 0.000E+00  
...  
"UPR-300-FF-1", 0.000E+00  
"UPR-600-12", 0.000E+00  
"US_Ecology", 3.673E+04  
"WRAP", 0.000E+00  
"All Sites", 1.957E+05  
2050,"year selected for output"  
482,"sites had no releases"  
497,"sites had releases"  
979,"total sites"
```

### 29.2.2.3 INVSUM Summary Statistics File

The INVSUM summary statistics file contains the summary statistics for the accumulated inventory for every waste site for the specified waste type and year. Table 29.3 contains excerpted records from an

INVSUM results file for the five analytes and the single waste form “core”. The file is written in comma-separated format for ease of importing into a spreadsheet program.

**Table 29.3** Excerpted Records from an INVSUM Summary Statistics File

"Program Name:", "InvSum"
"Program Version:", "1.1.001"
"Program Date:", "27 Oct 2005"
"Run ID:", "20051027161500"
"Keyword File Name:", "Test.key"
"Waste Forms:", "
"core"
"Analyte", "site", "Minimum", "5th Percentile", "10th Percentile", "25th
Percentile", "Median", "75th Percentile", "90th Percentile", "95th
Percentile", "Maximum", "Mean", "Standard Deviation"
"H3", "Sum_over_sites", 1.1886E+03, 1.3431E+03, 1.4144E+03, 1.5813E+03,
1.7685E+03, 1.9524E+03, 2.1182E+03, 2.2209E+03, 2.6584E+03, 1.7854E+03,
2.8382E+02
"Cl4", "Sum_over_sites", 3.7548E+04, 3.9306E+04, 4.0285E+04, 4.3206E+04,
4.6203E+04, 4.9511E+04, 5.1795E+04, 5.2541E+04, 5.7380E+04, 4.6307E+04,
4.3628E+03
"Cl36", "Sum_over_sites", 2.8205E+02, 3.0029E+02, 3.0611E+02, 3.1514E+02,
3.4653E+02, 3.6043E+02, 3.7401E+02, 3.9247E+02, 4.1220E+02, 3.4299E+02,
2.9123E+01
"Se79", "Sum_over_sites", 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00,
0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00,
0.0000E+00
"Sr90", "Sum_over_sites", 1.0084E+01, 1.1078E+01, 1.1427E+01, 1.1867E+01,
1.2748E+01, 1.3514E+01, 1.4138E+01, 1.4285E+01, 1.5166E+01, 1.2711E+01,
1.0811E+00

## 29.3 Keyword Definitions for INVSUM

In general, the keywords for the INVSUM code can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

### 29.3.1 ANALYTE Keyword for INVSUM

The ANALYTE keyword is used to define the analytes for which inventory data will be extracted. The following is this keyword’s syntax:

```
ANALYTE [ID="quote 1"] [FILE="quote 2"]
```

Analytes chosen must be a subset of the analytes used in the fate and transport activities controlled through the ESD keyword file. Multiple analyte keywords may be entered. A separate ANALYTE keyword is required for each desired analyte. The modifiers ID and FILE may be entered in any order. The modifiers associated with the ANALYTE keyword are described in Table 29.4.



**Table 29.4** Modifiers Associated with the ANALYTE Keyword in INVSUM

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte ID (up to six characters in length). The analyte ID entered must be one of the set of analytes identified in the environmental settings definition file.
FILE	The quote string associated with the FILE modifier is the name of the data file written by INGRAB for the specified analyte (file name up to 200 characters in length).

The following suite of ANALYTE keywords define three analytes for which inventory information is to be extracted:

```
ANALYTE ID="Tc99"   FILE="D:\SAC\Analyses\Init\Inv\INGRABA_TC_ALL.CSV"
ANALYTE ID="Sr90"   FILE="D:\SAC\Analyses\Init\Inv\INGRABA_SR_ALL.CSV"
ANALYTE ID="U238"   FILE="D:\SAC\Analyses\Init\Inv\INGRABA_U8_ALL.CSV"
```

### 29.3.2 END Keyword for INVSUM

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 29.3.3 FILE Keyword for INVSUM

The FILE keyword is used to enter the names of some input and output files. Other file names are defined using the ANALYTE and REPORT keywords. The following is this keyword's syntax:

```
FILE [ INDEX="quote1" ] { SUM="quote2" } { STATISTI="quote3" }
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code. Table 29.5 describes the modifiers associated with the FILE keyword.

**Table 29.5** Modifiers Associated with the FILE Keyword in INVSUM

Modifier	Description
INDEX	The input file identified in the quote string associated with the INDEX modifier contains lists of all of the times, locations, analytes, and waste types that were used in the INVENTORY code. Typically this file is named "inventory.all".
SUM	The output file identified in the quote string associated with the SUM modifier contains the primary output data for this run of the INVSUM code. A file extension of ".csv" is suggested. The data are comma separated and can be opened directly in a spreadsheet for plotting or other post-processing.
STATISTI	The output file identified in the quote string associated with the optional STATISTI modifier contains summary statistics for this run of the code. A file extension of ".csv" is suggested. The data are comma separated and can be opened directly in a spreadsheet for plotting or other post-processing. This option is most useful when the inventory run generated more than one realization of data.

The following three examples show the use of the FILE keyword:

```
FILE INDEX      "O:\History5-stoch\inventory\Inventory.All"  
FILE SUM        "Test_Sum.csv"  
FILE STATISTI  "Test_Sum_Sta.csv"
```

### 29.3.4 LIMITOUT Keyword for INVSUM

The optional LIMITOUT keyword is used to limit the amount of output to the report file and the results file. The following is this keyword's syntax:

```
LIMITOUT [EACH] [MAX]
```

If the modifier EACH is entered, the output of summary information from each input file is suppressed. If the modifier MAX is entered, the output of maximum information across all analytes is suppressed. If both modifiers are entered, no useful summary information will be output. The following two examples show the use of the LIMITOUT keyword:

```
LIMITOUT EACH  
LIMITOUT MAX
```

### 29.3.5 LIMITSIT Keyword for INVSUM

The optional LIMITSIT keyword is used to limit the output to a user-specified list of waste sites instead of every waste site in the input files. The following is this keyword's syntax:

```
LIMITSIT {"quote1"} ... {"quoten"}
```

If this keyword is used, information will only be output for the waste sites identified in one of the quote strings. The other waste sites will be ignored. This keyword is useful if information is desired for only one site or a small group of sites. The following example shows the use of the LIMITSIT keyword:

```
LIMITSITE "218-E-RCRA" "US_Ecology" "YUCCA_H2-HLW-CANISTER"
```

### 29.3.6 REPORT Keyword for INVSUM

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 29.3.7 STATISTI Keyword for INVSUM

The presence of the optional STATISTICS keyword initiates calculation of summary statistics for each of the site sums and sums over multiple sites. These values are output to a file identified using the FILE keyword with the STATISTI modifier. The following is this keyword's syntax:

```
STATISTI
```

### 29.3.8 TITLE Keyword for INVSUM

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the INVSUM code."
```

### 29.3.9 USER Keyword for INVSUM

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 29.3.10 VERBOSE Keyword for INVSUM

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. The following is this keyword's syntax:

```
VERBOSE
```

### 29.3.11 WASTE Keyword for INVSUM

The WASTE keyword is used to identify one or more waste streams for which data will be extracted. The program will error terminate unless a single WASTE keyword is used. The following is this keyword's syntax:

```
WASTE [ "quote" ] ... { "quote" }
```

The waste stream IDs are entered in quote strings, which must be enclosed in double quotation marks. Waste stream IDs up to 24 characters long are supported. The following example defines a code run that extracts data for three waste streams:

```
WASTE "cmntcs" "cement" "soillf"
```

A typical use of INVSUM is to extract inventory data summed over all waste types. In that case, the ID “All Types” should be used.

### **29.3.12 YEAR Keyword for INVSUM**

The YEAR keyword is used to identify the single year for which data will be extracted. The program will error terminate if the year is not supplied. The following is this keyword’s syntax:

```
YEAR [number]
```

The year is entered as an integer calendar year and should be in the range of years identified in the environmental settings definition keyword file. The following example defines a code run that extracts inventory data for the year 2050:

```
YEAR 2050
```

## 30.0 L3IDIFF – Differencing CFEST Nodal Fluid Sources

### 30.1 Overview

L3IDIFF is a utility code for differencing nodal fluid sources in two otherwise identical CFEST *.l3i* input files. This utility is expressly designed to support the calculation of artificial fluid discharges by permitting a vadose site or sites to be modeled twice, once with artificial fluid sources and once without, and using this tool to find the portion of the fluid discharged from the vadose zone attributable to artificial sources only.

L3IDIFF checks every line read from the two input *.l3i* files (“natural.l3i” and “total.l3i”) to ensure they match exactly, and produces a fatal error if they do not. The only exception is the lines that define nodal sources, as these are expected to vary, reflecting differing liquid discharges for the “natural” and “total” signals. The two input *.l3i* files are opened as “read” only and will not be altered by L3IDIFF.

#### 30.1.1 Location in the Processing Sequence

L3IDIFF is designed to be executed after completely simulating the vadose zone twice (once with the standard SAC inventory “.res” file, once with an empty inventory “.res” file to remove all artificial fluid sources).

#### 30.1.2 How the Code Is Invoked

L3IDIFF is invoked with the following command-line calling arguments:

```
l3idiff-1.exe natural.l3i total.l3i artificial.l3i
```

L3IDIFF is expected to be used only as a standalone utility program and will not be invoked by any other SAC program.

#### 30.1.3 Memory Requirements

The L3IDIFF code uses dynamic allocation of memory. Memory requirements are proportional to the number of time steps in the CFEST *.l3i* files being processed.

## 30.2 File Definitions

The L3IDIFF code reads two files and writes two files.

### 30.2.1 Input Files

The L3IDIFF code reads two *.l3i* files, one representing the “natural” signal and one the “total” signal (natural plus artificial fluid sources).

The file name for the natural signal is “natural.l3i”. This file is the CFEST *.l3i* input file, prepared with no SAC artificial fluid discharges (i.e., result of VZDROP run against a “vzdrop.key” file pointing to STOMP “discharge” files for all vadose sites simulated with an empty “inv\*.res” file).

The file name for the total signal is “total.l3i”. This file is the CFEST *.l3i* input file, prepared with SAC artificial fluid discharges (i.e., result of VZDROP run against a “vzdrop.key” file pointing to STOMP “discharge” files for all vadose sites simulated with a normal “inv\*.res” file).

### **30.2.2 Output Files**

Two files are written by the L3IDIFF code: an “artificial” *.l3i* file that contains the difference between the nodal sources in the “total” and “natural” *.l3i* files, and a log file for the run called “l3idiff.log”. The log file contains run status reports and error messages.

The file name for the output l3i file is “artificial.l3i”. This file is a CFEST *.l3i* input file to be generated by this utility (L3IDIFF) that will contain nodal sources and sinks that are attributable only to the artificial discharge component of all fluid influxes to the vadose zone.

## **31.0 MAKEU – Inventory Data Uranium Conversion**

### **31.1 Overview**

This program reads one or more inventory results (\*.res) files generated by the SAC Rev. 1 inventory code and outputs a revised results file in the same format. The revision is to add a new analyte (uranium as an element) that is computed as a weighted sum of all uranium isotopes.

#### **31.1.1 Location in the Processing Sequence**

MAKEU reads files produced by the inventory code. It can be run anytime after the inventory code completes.

#### **31.1.2 How the Code Is Invoked**

MAKEU can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), MAKEU executes in a DOS box. A run of MAKEU is initiated by entering the following command line:

```
MAKEU "Keyfilename"
```

Under the Linux operating system MAKEU is executed through any of the following Bourne Shell or C Shell commands:

```
makeu-1.exe "Keyfilename"
```

For these commands, “MAKEU.EXE” or “makeu-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If MAKEU is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If MAKEU cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **31.1.3 Memory Requirements**

The memory used by MAKEU program depends on the number of sites and waste streams. A production run of the code using 1000 waste sites and 40 waste streams is expected to take less than 9 MB of memory.

### **31.2 File Definitions**

The MAKEU program reads two or more input files and writes two or more output files. These files are described in the following sections.

### 31.2.1 Input Files

The MAKEU program reads a control keyword file and a one or more results file written by the INVENTORY code. The MAKEU keyword file contains control information. An example file is provided in Table 31.1. Detailed definitions of the keywords are provided in Section 31.3.

**Table 31.1** Example Keyword File for MAKEU

```
FILE REPORT "MakeU.rpt"
TITLE "Inventory Sums by Site to convert Uranium Isotopes to Mass"
USER "Paul W. Eslinger"
!
!FILE EXPLICIT "inv486.res"
FILE REALIZAT FROM 1 TO 500 BASIS 500
!
OS "Unix"
!
OUTPUT ID="U" UNITS="Kg"
SUM ID="U233" FACTOR= 1.03688E-01 UNITS="Ci"
SUM ID="U234" FACTOR= 1.60584E-01 UNITS="Ci"
SUM ID="U235" FACTOR= 4.62468E+02 UNITS="Ci"
SUM ID="U238" FACTOR= 2.97397E+03 UNITS="Ci"
!
END
```

### 31.2.2 Output Files

The MAKEU program writes two or more output files. One file is a report file that contains text information about the run of the code. It is not described further here; however, it should always be examined after a run of the code because error messages are written to the report file. The other output file is a modified results (\*.res) file. This file has the uranium mass (kg) entered at the end of the data for every waste site that contains one or more uranium isotopes. Data for one liquid waste stream for the site 100-B-5 is shown in Table 31.2.

**Table 31.2** Excerpt from a MAKEU-generated Results File

```
"100-B-5",3,"Onsite location, number of years of data that follow"
1954,1,8,"m^3","liquid", 2.41930E+04
"H3","Ci", 2.03824E-04
"Tc99","Ci", 2.09724E-05
"I129","Ci", 1.25891E-07
"U233","Ci", 0.00000E+00
"U234","Ci", 8.75423E-09
"U235","Ci", 7.99542E-06
"U238","Ci", 1.34470E-04
"U","Kg", 4.03607E-01
```



### 31.3 Keyword Definitions for MAKEU

In general, the keywords for the MAKEU code can be entered in any order. The only restriction is that the END keyword must be the last entry.

#### 31.3.1 END Keyword for MAKEU

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

#### 31.3.2 FILE Keyword for MAKEU

The FILE keyword is used to define the names of input and output files. There are two general forms for this keyword's syntax:

```
FILE [modifier="quote 1"]  
FILE [REALIZAT] [FROM=N1] [TO=N2] [BASIS=N3]
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. Only one file can be defined with each entry of the file keyword. At least one FILE keyword is required for every run of the code. The modifiers associated with the FILE keyword are given in Table 31.3.

**Table 31.3** Modifiers Associated with the FILE Keyword in MAKEU Keyword File

Modifier	Description
REPORT	The quote string associated with the REPORT modifier defines the name of the report (log) file.
EXPLICIT	The quote string associated with the EXPLICIT modifier defines the name of the inventory results file that will be modified. This version of the file keyword should not be used when the REALIZAT modifier is used.
REALIZAT	The REALIZAT modifier instructs the code to develop default file names for one or more inventory results files. The numerical values associated with the FROM and TO modifiers identifies the range of realizations for which inventory data will be partitioned. The numerical value associated with the BASIS modifier identifies the largest realization number in the inventory run that generated the results files. This version of the file keyword should not be used when the EXPLICIT modifier is used.

The report file for a run of the code could be defined using the following keyword:

```
FILE REPORT "MAKEU.rpt"
```

The following keyword would cause the MAKEU code to operate only on one explicitly named inventory results file.

```
FILE EXPLICIT "inv1.res"
```

The following keyword would cause the MAKEU code to operate on the separate files containing inventory in realizations 1 through 100 that were generated by a run from the inventory code that processed exactly 100 realizations.

```
FILE REALIZAT FROM 1 TO 100 BASIS 100
```

### 31.3.3 OS Keyword for MAKEU

The OS keyword is used to define the operating system under which the code is executed. The following is this keyword's syntax:

```
OS ["Unix" | "Windows"]
```

There is no default for the OS keyword. One of the operating systems must be chosen. The following keyword selects the Unix (Linux) operating system

```
OS "Unix"
```

### 31.3.4 OUTPUT Keyword for MAKEU

The OUTPUT keyword is used to define the output analyte ID and the data units for the output analyte. The following is this keyword's syntax:

```
OUTPUT [ID "quote"] [UNITS="quote2"]
```

The quote string associated with the ID modifier contains the ID for the output analyte. The quote string associated with the UNITS modifier identifies the units for the output analyte. Nominally, as illustrated in the following keyword, the output analyte ID is "U" and the data units are "Kg."

```
OUTPUT ID="U" UNITS="Kg"
```

The SAC codes implicitly assume that all mass units for hazardous chemicals are kilograms. Entering an output units string that is not "Kg" may lead to later computational difficulties.

### 31.3.5 SUM Keyword for MAKEU

The SUM keyword is used to define the suite of analytes that will be converted to mass units and summed. The following is this keyword's syntax:

```
SUM [ID "quote"] [FACTOR=N1] [UNITS="quote2"]
```

The quote string associated with the ID modifier contains the ID for the input analyte. The numerical value associated with the FACTOR modifier contains the multiplicative conversion factor (units are kg/Ci). The conversion factor is the multiplier that converts Curies of the isotope to kilograms of the

element. The quote string associated with the UNITS modifier identifies the data units for the input data. The SAC modeling system makes the implicit assumption that radioactive isotopes have units of Ci. Entering any units other than Ci may cause processing difficulties in the modeling system.

Multiple analytes can be used in the sum. Each analyte in the sum must be defined with a separate SUM keyword. A set of five keywords that will compute uranium mass from five uranium isotopes are the following:

```
SUM ID="U233" FACTOR= 1.0369E-01 UNITS="Ci "  
SUM ID="U234" FACTOR= 1.6072E-01 UNITS="Ci "  
SUM ID="U235" FACTOR= 4.6247E+02 UNITS="Ci "  
SUM ID="U236" FACTOR= 1.5448E+01 UNITS="Ci "  
SUM ID="U238" FACTOR= 2.9740E+03 UNITS="Ci "
```

### 31.3.6 TITLE Keyword for MAKEU

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the MAKEU code."
```

### 31.3.7 USER Keyword for MAKEU

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 31.3.8 VERBOSE Keyword for MAKEU

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```



## 32.0 PICK\_HYDRO – Constrained Vadose Zone Parameters

### 32.1 Overview

The PICK\_HYDRO program contains the logic to define constrained sampling for the parameters in the vadose zone flow model (STOMP code). This logic is only invoked when random sampling from the statistical distributions for the parameters result in unreasonable flow or transport results due to unrealistic combinations of the input parameters. This code defines the constraints to use in the sampling process. The ESP code implements the constraints when it samples the vadose zone hydraulic model parameters and uses them to populate STOMP input files.

#### 32.1.1 Algorithms

The vadose zone flow model (STOMP code) uses the van Genuchten formulation (van Genuchten 1980) for the effective aqueous-phase saturation in conjunction with the Mualem formulation (Mualem 1976) for the aqueous-phase relative permeability. A total of five empirically derived parameters are needed to fully define the resulting curves for effective aqueous-phase saturation and effective aqueous-phase permeability. The effective aqueous-phase saturation  $s_\ell^*$  is defined as

$$s_\ell^* = \frac{s_\ell - s_{\ell r}}{1 - s_{\ell r}}$$

where  $s_\ell$  is the aqueous-phase saturation and  $s_{\ell r}$  is the residual aqueous-phase saturation. The van Genuchten formulation for the aqueous-phase saturation is defined as follows:

$$\begin{aligned} s_\ell^* &= \left[ 1 + (\alpha \psi)^n \right]^{-(1-\frac{1}{n})} & \text{for } \psi < 0 \text{ (unsaturated conditions)} \\ s_\ell^* &= 1 & \text{for } \psi \geq 0 \text{ (saturated conditions)} \end{aligned}$$

where  $\psi$  is the tension head and  $\alpha$  and  $n$  are the empirically determined fitting parameters. In practice, this curve is multiplied by the porosity of the hydrologic unit. The aqueous-phase relative permeability ( $k_{r\ell}$ ) calculations use the Mualem formulation:

$$k_{r\ell} = \sqrt{s_\ell^*} \left[ 1 - \left( 1 - (s_\ell^*)^{\frac{1}{m}} \right)^m \right]^2.$$

The effective permeability ( $k_\ell$ ) is computed as  $k_\ell = K_s k_{r\ell}$  where  $K_s$  is the saturated aqueous-phase permeability. The parameter  $m$  is defined using the expression  $m = 1 - 1/n$ . The five sampled parameters are  $\alpha$ ,  $n$ , porosity,  $K_s$  and  $s_{\ell r}$ .

The PICK\_HYDRO program starts from a reasonable set of input parameters, called the basis parameters, to choose constraints on the combinations of stochastically sampled parameters. A sample-and-reject technique is used to ensure sampled combinations of sets of five parameters yield curves for saturation and permeability that are reasonably close to the curves obtained using the basis parameters. The sampling technique consists of the following steps:

1. Obtain a sample of all five parameters from the parameter distributions derived from field sampling data, where the distributions are for each parameter individually.

2. Compute the tension head versus aqueous-phase curve using the sampled parameters.
3. Compute the integrated distance between the curve in step 2 and a similar curve calculated using the basis parameter values.
4. Compute the effective permeability versus saturation curve using the sampled parameters.
5. Compute the integrated distance between the curve in step 4 and a similar curve calculated using the basis parameter values.
6. Reject the set of five sampled values if either of the distances from steps 3 and 5 were larger than user supplied limits.

### 32.1.2 Location in the Processing Sequence

The PICK\_HYDRO program prepares data that are used in the setup for vadose zone flow and transport. Therefore, it must be run at the very start of an analysis.

### 32.1.3 How the Code Is Invoked

PICK\_HYDRO can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), PICK\_HYDRO executes in a DOS box. A run of PICK\_HYDRO is initiated by entering the following command line:

```
PICK_HYDRO "Keyfilename"
```

Under the Linux operating system PICK\_HYDRO is executed through any of the following Bourne Shell or C Shell commands:

```
pick_hydro.exe "Keyfilename"
```

For these commands, "PICK\_HYDRO.EXE" or "pick\_hydro.exe" is the name of the executable program, and "Keyfilename" is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If PICK\_HYDRO is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If PICK\_HYDRO cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

### 32.1.4 Memory Requirements

The PICK\_HYDRO code uses dynamic memory allocation. It is expected that all of the runs of the PICK\_HYDRO code will require less than 5 MB of memory.

## 32.2 File Definitions

The PICK\_HYDRO code reads one input file and writes five output files.

### 32.2.1 Input Files

The PICK\_HYDRO code reads a single input file containing control keywords. An example keyword file is provided in Table 32.1.

**Table 32.1** Example Keyword File for PICK\_HYDRO

```
REPORT "Pick_History2_VSlight.rpt"
USER "Paul W. Eslinger"
TITLE "Develop vadose zone parameters - Slightly Constrained"
! Number of samples to generate
SAMPLES REALIZAT=2000 TRIES=10000
!GOOD
!VERBOSE
SEED 2324688.
RELAX DELV=1.25 DELM=1.25 ITER=10
!
FILE VANG "Pick_History2_VSlight_vang.csv" !ALL
FILE MUAL "Pick_History2_VSlight_mual.csv" !ALL
FILE SAMP "Pick_History2_VSlight_samp.csv" !ALL
FILE ROCK "Pick_History2_VSlight_basis.key"
ENDINIT
!
! Rock Types (Cases) - Multiple cases can be entered
CASE
TITLE="Hss (Hanford silty fine sand)"
STOCHASTIC ID="THET" 7 4.45E-01 5.99E-02 ! Total Porosity
STOCHASTIC ID="RSAT" 7 1.58E-01 5.91E-02 ! Saturation Residual
STOCHASTIC ID="VANA" 9 -4.86E+00 1.21E+00 ! Saturation Air
STOCHASTIC ID="VANN" 7 1.91E+00 4.61E-01 ! Saturation Exponent
STOCHASTIC ID="KSAT" 9 -9.36E+00 1.88E+00 ! Hydraulic
BASIS THET=4.46E-01 RSAT=1.60E-01 VANA=8.06E-03 VANN=1.95 KSAT=6.67E-05
METRIC_V M_MIN=1.0E-1 M_MAX=100000 LIMIT=1.49000E+01 STEPS=250
METRIC_M M_MIN=.1 M_MAX=.999995 LIMIT=6.45000E-04 STEPS=250
ITERATE DELV=0.02 DELM=0.0000025 STEPS=200 REJECT=4000
ENDCASE
!
END
```

### 32.2.2 Output Files

The PICK\_HYDRO program writes five output files:

- **Report File:** The report contains text information about the run of the code. It should always be examined after a run of the code because error messages are written to the report file.
- **Rock Basis Data:** The rock basis data file contains keyword data for the ESP code for multiple rock types that define the constraints to be used for the vadose zone. This file contains the primary output of the code that is by the ESP code in setting up input data for the vadose zone flow and transport model. An example rock basis file for four rock types is provided in Table 32.2

**Table 32.2** Example Rock Basis Keyword File Output from the PICK\_HYDRO Code

```
! Data Traceability
!
! Program : Pick_Hydro
! Version : 1.02.001
```

```
! Modified: 7 Mar 2006
! Run ID : 20060308093232
! User : Paul W. Eslinger
! Run Date: 03/08/2006
! Run Time: 09:32:32.115
! Key File: Pick_History2_VSlight.key
!
BASIS ROCK="B" KSAT= 9.1439E-04 VANA= 1.8526E-02
  VANN= 1.4156E+00 THET= 2.7011E-01 RSAT= 1.0505E-01
  VanMin= 1.0000E-01 VanMax= 1.0000E+05 VanLim= 9.6400E+00
  MualMin= 1.0000E-01 MualMax= 9.99995E-01 MualLim= 3.31450E-03 Steps=250
!
BASIS ROCK="Hss" KSAT= 6.6759E-05 VANA= 8.0680E-03
  VANN= 1.9593E+00 THET= 4.4613E-01 RSAT= 1.6086E-01
  VanMin= 1.0000E-01 VanMax= 1.0000E+05 VanLim= 1.4900E+01
  MualMin= 1.0000E-01 MualMax= 9.99995E-01 MualLim= 6.45000E-04 Steps=250
!
BASIS ROCK="Hss_2W" KSAT= 2.1636E-05 VANA= 4.3733E-03
  VANN= 2.1940E+00 THET= 4.1111E-01 RSAT= 1.3818E-01
  VanMin= 1.0000E-01 VanMax= 1.0000E+05 VanLim= 1.2240E+01
  MualMin= 1.0000E-01 MualMax= 9.99995E-01 MualLim= 1.80000E-04 Steps=250
!
BASIS ROCK="Hss_U" KSAT= 2.4941E-05 VANA= 6.8535E-03
  VANN= 2.3637E+00 THET= 4.4573E-01 RSAT= 1.4045E-01
  VanMin= 1.0000E-01 VanMax= 1.0000E+05 VanLim= 1.1200E+01
  MualMin= 1.0000E-01 MualMax= 9.99995E-01 MualLim= 2.60000E-04 Steps=250
```

- **van Genuchten Data:** This output file contains plot information about the effective aqueous-phase saturation versus tension head (van Genuchten formulation) for each rock type. The basis curve is always output and sampled curves can also be output under control of the GOOD keyword or the FILE keyword.
- **Mualem Data:** This output file contains plot information about the relative permeability versus saturation curve (Mualem formulation) for each rock type. The basis curve is always output and sampled curves can also be output under control of the GOOD keyword or the FILE keyword.
- **Sampled Data:** This output file is always created. It contains good samples or rejected samples (or both) of the five sampled parameters for each rock type depending on the output control of the GOOD keyword and the FILE keywords.

### 32.3 Keyword Definitions for PICK\_HYDRO

The keywords for the PICK\_HYDRO code are entered in two groups. In general, the keywords within a group can be entered in any order. The first group must start with the REPORT keyword and end with the ENDINIT keyword. The second group of keywords defines one or more sampling cases (each starting with a CASE keyword and ending with an ENDCASE keyword). The keyword file always ends with the END keyword.



### 32.3.1 BASIS Keyword for PICK\_HYDRO

The BASIS keyword used within a CASE/ENDCASE construct. It is used to enter the basis values for the five sampled parameters for a single rock type. The following is this keyword's syntax:

```
BASIS [THET=N1] [RSAT=N2] [VANA=N3] [VANN=N4] [KSAT=N5]
```

The modifiers associated with the BASIS keyword can be entered in any order and are defined in Table 32.3.

**Table 32.3** Modifiers Associated with the BASIS Keyword in PICK\_HYDRO

Modifier	Description
KSAT	The numerical value associated with the KSAT modifier provides the saturated aqueous-phase permeability for the rock type. Units are cm/s.
RSAT	The numerical value associated with the RSAT modifier is the residual saturation of the rock type. This value is unitless.
THET	The numerical value associated with the THET modifier is the total porosity of the rock type. This value is unitless.
VANA	The numerical value associated with the VANA modifier is the alpha parameter in the van Genuchten formulation. Units are $\text{cm}^{-1}$ .
VANN	The numerical value associated with the VANN modifier is the exponent parameter in the van Genuchten formulation. This value is unitless.

An example use of this keyword is the following:

```
BASIS THET=2.7011E-01 RSAT=1.0505E-01 VANA=1.8526E-02  
VANN=1.4156E+00 KSAT=9.1439E-04
```

### 32.3.2 CASE Keyword for PICK\_HYDRO

The CASE keyword is used to start the definition of parameters for a single rock type in the second group of keywords. It is always paired with an ENDCASE keyword. Multiple CASE keywords can be entered. The following is this keyword's syntax:

```
CASE
```

No other information is entered on the CASE keyword.

### 32.3.3 END Keyword for PICK\_HYDRO

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 32.3.4 ENDCASE Keyword for PICK\_HYDRO

The ENDCASE keyword is used to signify the end of definition of parameters for a single rock type in the second group of keywords. It is always paired with a CASE keyword. The following is this keyword's syntax:

```
ENDCASE
```

No other information is entered on the ENDCASE keyword.

### 32.3.5 ENDINIT Keyword for PICK\_HYDRO

The ENDINIT keyword is used to signify the end of the initial set of keywords. The initial set of keywords define information that is applicable to all rock types. The following is this keyword's syntax:

```
ENDINIT
```

No other information is entered on the ENDINIT keyword.

### 32.3.6 FILE Keyword for PICK\_HYDRO

The FILE keyword is used to define the names of input and output files. This keyword must be entered in the initial set of keywords. The following is this keyword's syntax:

```
FILE MODIFIER "Quote 1" {ALL}
```

The modifiers associated with the FILE keyword can be entered in any order and are defined in Table 32.4. Multiple FILE keywords must be entered because only one file can be defined at a time.

**Table 32.4** Modifiers Associated with the FILE Keyword in PICK\_HYDRO

Modifier	Description
MUAL	The quote string associated with the MUAL modifier defines the name of the output file that will contain data for the Mualem formulation of relative permeability versus saturation. All generated curves will be written to the file for all rock types if the modifier ALL is included. Otherwise, only the basis curve is output for every rock type. The presence of the ALL modifier also initiates output of the sampled values to the sample file.
ROCK	The quote string associated with the ROCK modifier defines the name of the output file that will contain keyword data used in the ESP program to control the sampling process. This data file contains the primary output data from the code.
SAMP	The quote string associated with the SAMP modifier defines the name of the output file that will contain sampled values for all generated data. Even though the file is created for every run of the code, generated samples are written to the file only if the optional modifier ALL is included.
VANG	The quote string associated with the VANG modifier contains the name of the output file that will contain data for the van Genuchten formulation of effective aqueous-phase saturation versus tension head. All generated curves will be written to the file for all rock types if the modifier ALL is included. Otherwise, only the basis curve is output for every rock type. The presence of the ALL modifier also initiates output of the sampled values to the sample file.

Example uses of the FILE keyword for defining four output files is the following:

```
FILE VANG "Pick_History2_VTight_vang.csv" ALL
FILE MUAL "Pick_History2_VTight_mual.csv" ALL
FILE SAMP "Pick_History2_VTight_samp.csv" ALL
FILE ROCK "Pick_History2_VTight_basis.key"
```

### 32.3.7 GOOD Keyword for PICK\_HYDRO

The GOOD keyword is used to turn on a flag to signal output of plotting data or sampled values for only good realizations. This keyword has no modifiers. This keyword must be entered in the initial set of keywords. The following is this keyword's syntax:

```
GOOD
```

No other information is entered on the GOOD keyword.

### 32.3.8 ITERATE Keyword for PICK\_HYDRO

The optional ITERATE keyword invokes an iterative scheme intended to improve the selection of the limit values defined using the METRIC\_M and METRIC\_V keywords. This keyword must be entered within a CASE/ENDCASE set of keywords. The following is this keyword's syntax:

```
ITERATE [DELV=N1] [DELM=N2] [STEPS=N3] [REJECT=N4]
```

The iterative scheme tries to revise limits for both the van Genuchten and Mualem curves to achieve the same number of sample rejections from the van Genuchten curve as from the Mualem curve.

Convergence is declared when the target number of rejections is within 3% of the desired number and the ratio of rejections from the two curves is within 5% of unity. The modifiers for this keyword can be entered in any order and are described in Table 32.5

**Table 32.5** Modifiers Associated with the ITERATE Keyword in PICK\_HYDRO

Modifier	Description
DELV	The numerical value associated with the DELV modifier is the size of the step to take in the van Genuchten limit when attempting to meet the convergence criteria. An initial value of approximately 1% of the limit is suggested.
DELM	The numerical value associated with the DELM modifier is the size of the step to take in the Mualem limit when attempting to meet the convergence criteria. An initial value of approximately 1% of the limit is suggested.
REJECT	The numerical value associated with the REJECT modifier is the target number of samples to reject when attempting to meet the convergence criteria. This value must be selected in concert with the entries on the SAMPLES keyword. Typically it is set to the difference of the number of tries and the number of realizations.
STEPS	The numerical value associated with the STEPS modifier identifies the maximum number of iteration steps to try before declaring failed convergence and moving on to the next CASE definition.

An example use of this keyword is the following:

```
ITERATE DELV=0.01 DELM=0.00000025 STEPS=200 REJECT=38000
```

### 32.3.9 METRIC\_M Keyword for PICK\_HYDRO

The METRIC\_M keyword is used to define the value of the metric for determining a proper selection of data for the Mualem curve. The metric is a sum of differences of equally spaced points (in log space) between the base relative permeability versus saturation curve and the same curve calculated using sampled parameter values. This keyword must be entered within a CASE/ENDCASE set of keywords. The following is this keyword's syntax:

```
METRIC_M [M_MIN=N1] [M_MAX=N2] [LIMIT=N3] [STEPS=N4]
```

The modifiers for this keyword can be entered in any order and are described in Table 32.6.

**Table 32.6** Modifiers Associated with the METRIC\_M Keyword in PICK\_HYDRO

Modifier	Description
LIMIT	The Mualem curve based on a sampled set of parameters is considered a valid curve if the distance metric is less than the numerical value associated with the LIMIT keyword. A value greater than zero is required. A value of 0.001 is typical.
M_MIN	The numerical value associated with the M_MIN modifier is the minimum moisture content to consider. A value greater than zero is required. A value of 0.1 is typical.
M_MAX	The numerical value associated with the M_MAX modifier is the maximum moisture content to consider. A positive value greater than the minimum value is required. A value of 1.0 is typical.
STEPS	The numerical value associated with the STEPS modifier defines the number of steps to use in calculating the difference sum. A value of 250 is recommended. The same value should be used on this keyword and on the METRIC_V keyword.

An example use of this keyword is the following:

```
METRIC_M M_MIN=0.1 M_MAX=.999995 LIMIT=2.02500E-04 STEPS=250
```

### 32.3.10 METRIC\_V Keyword for PICK\_HYDRO

The METRIC\_V keyword is used to define the value of the metric for determining a proper selection of data for the van Genuchten curve. The metric is a sum of differences of equally spaced points (in log space) between the base aqueous phase saturation and tension head curve and the same curve calculated using sampled parameter values. This keyword must be entered within a CASE/ENDCASE set of keywords. The following is this keyword's syntax:

```
METRIC_V [M_MIN=N1] [M_MAX=N2] [LIMIT=N3] [STEPS=N4]
```

The modifiers for this keyword can be entered in any order and are described in Table 32.7.

**Table 32.7** Modifiers Associated with the METRIC\_V Keyword in PICK\_HYDRO

Modifier	Description
LIMIT	The van Genuchten curve based on a sampled set of parameters is considered a valid curve if the distance metric is less than the numerical value associated with the LIMIT keyword. A value greater than zero is required. A value of 10 is typical.
M_MIN	The numerical value associated with the M_MIN modifier is the minimum tension head to consider. A value greater than zero is required. A value of 0.1 is typical. Units for this entry are cm.
M_MAX	The numerical value associated with the M_MAX modifier is the maximum tension head to consider. A value greater than zero is required. A value of 100,000 is typical. Units for this entry are cm.
STEPS	The numerical value associated with the STEPS modifier defines the number of steps to use in calculating the difference sum. A value of 250 is recommended. The same value should be used on this keyword and on the METRIC_M keyword.

An example use of this keyword is the following:

```
METRIC_V M_MIN=1.0E-1 M_MAX=100000 LIMIT=1.23900E+01 STEPS=250
```

### 32.3.11 RELAX Keyword for PICK\_HYDRO

The optional RELAX keyword is associated with the iterative scheme invoked by the ITERATE keyword to improve the selection of the limit values input using the METRIC\_M and METRIC\_V keywords. It only has an effect for those cases where an ITERATE keyword is entered. This keyword must be entered in the initial set of keywords and applies to all ITERATE keywords. The following is this keyword's syntax:

```
RELAX [DELV=N1] [DELM=N2] [ITER=N3]
```

The purpose of this keyword is to relax convergence criteria on the iterative approach. This option sometimes helps determine an approximate set of limits that can then be refined using the iterative method without relaxation. The modifiers associated with the RELAX keyword can be entered in any order and are defined in Table 32.8.

**Table 32.8** Modifiers Associated with the RELAX Keyword in PICK\_HYDRO

Modifier	Description
DELV	The numerical value associated with the DELV modifier is the multiplier for relaxing the iterative delta step for the van Genuchten curve provided on an ITERATE keyword. A value greater than 1 is needed, and a value of 1.25 is suggested.
DELM	The numerical value associated with the DELM modifier is the multiplier for relaxing the iterative delta step for the Mualem curve provided on an ITERATE keyword. A value greater than 1 is needed, and a value of 1.25 is suggested.
ITER	The numerical value associated with the ITER modifier is the number of steps to wait before applying the relaxation factors. For example, a value of 5 indicates to apply the relaxation factors every fifth iterative step.

An example use of this keyword that increases the limits for both the van Genuchten and Mualem curves by 25% every 5 iterative steps is the following:

```
RELAX DELV=1.25 DELM=1.25 ITER=5
```

### 32.3.12 REPORT Keyword for PICK\_HYDRO

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT "/SAC/C/Test.rpt"
```

### 32.3.13 SAMPLES Keyword for PICK\_HYDRO

The SAMPLES keyword defines the number of successful realizations that are needed and provides the maximum number of realizations that can be considered. This keyword must be entered in the initial set of keywords. The following is this keyword's syntax:

```
SAMPLES [REALIZAT=N1] [TRIES=N2]
```

The numerical value associated with the REALIZAT modifier defines the number of realizations of samples that meet the convergence criteria that are desired. The numerical value associated with the TRIES modifier defines the maximum number of realizations that can be considered in the sample-and-reject scheme while trying to achieve a given number of successful realizations. An example use of this keyword where up to 60,000 tries are allowed in attempting to achieve 2,000 good sample sets is the following:

```
SAMPLES REALIZAT=2000 TRIES=60000
```

### 32.3.14 SEED Keyword for PICK\_HYDRO

The SEED keyword sets the value for the seed for the random number generator. This keyword must be entered in the initial set of keywords. The following is this keyword's syntax:

```
SEED Value1
```

The value for Value1 must be an integer or real number in the range 1 to 999999. The following is an example keyword record:

```
SEED 344443
```

### 32.3.15 STOCHASTIC Keyword for PICK\_HYDRO

The STOCHASTIC keyword is used to enter the definition of a statistical distribution for stochastic variables. The following is this keyword's syntax:

```
STOCHASTIC ["Quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2} {"Quote2"}
```

- The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive, and embedded spaces are significant.
- The optional entry for Quote2 is a description for the stochastic variable that can be up to 64 characters long.
- The entry for Dist\_Index must be an integer in the range 1 to 13 that identifies the index of a statistical distribution. The available statistical distributions are defined in Table 45.1, and further information on statistical algorithms is provided in Section 45.0.
- The word Parameters in the general syntax statement indicates the numerical values of one or more parameters required for defining the statistical distribution.
- The additional modifier TRUNCATE can be used for all distribution types except 1, 3, and 10 (constant, discrete uniform, and user-defined). If TRUNCATE is entered, it must be followed by two values in the interval 0 to 1, inclusive. The lower value must be less than the upper value. These two values specify the tail probabilities at which to impose range truncation for the distribution. Truncation data must be entered after all of the other parameters that define the distribution.

Five stochastic keywords are needed for every CASE definition. The same five stochastic variable IDs are used in every CASE statement, and these are defined in Table 32.9.

**Table 32.9** Variable IDs Needed for the STOCHASTIC Keyword in PICK\_HYDRO

ID	Description
KSAT	This variable defines the statistical distribution for the saturated aqueous-phase permeability for the rock type. Units are cm/s.
RSAT	This variable defines the statistical distribution for the residual saturation of the rock type. This value is unitless.
THET	This variable defines the statistical distribution for the total porosity of the rock type. This value is unitless.
VANA	This variable defines the statistical distribution for the alpha parameter in the van Genuchten formulation. Units are cm <sup>-1</sup> .
VANN	This variable defines the statistical distribution for the exponent parameter in the van Genuchten formulation. This value is unitless.

The following set of five stochastic keywords provides examples of all the stochastic distributions needed for a single rock type:

```
STOCHASTIC ID="THET" 7 4.4540E-01 5.9995E-02 TRUNCATE 1.89E-02 9.90E-01
STOCHASTIC ID="RSAT" 7 1.5875E-01 5.9122E-02 TRUNCATE 2.96E-02 9.98E-01
STOCHASTIC ID="VANA" 9 -4.8657E+00 1.2119E+00 TRUNCATE 3.08E-02 9.99E-01
STOCHASTIC ID="VANN" 7 1.9150E+00 4.6127E-01 TRUNCATE 7.82E-02 9.98E-01
STOCHASTIC ID="KSAT" 9 -9.3631E+00 1.8850E+00 TRUNCATE 1.50E-03 8.92E-01
```

### 32.3.16 TITLE Keyword for PICK\_HYDRO

The TITLE keyword is used to define a single-line problem title. The problem title will be written to the report file. This keyword must be entered in the initial set of keywords. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the PICK_HYDRO code."
```

### 32.3.17 USER Keyword for PICK\_HYDRO

The USER keyword is used to identify the user of the program. The user name will be written to the report file. This keyword must be entered in the initial set of keywords. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 32.3.18 VERBOSE Keyword for PICK\_HYDRO

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. This keyword must be entered in the initial set of keywords. The following is this keyword's syntax:

```
VERBOSE
```

No other information is entered on the VERBOSE keyword.



## 33.0 PRESTOMP – STOMP Input File Modification (timing info)

### 33.1 Overview

PRESTOMP is a preprocessor utility code for STOMP, used either by ESP or independently, that provides the following functions for refining a STOMP input file before STOMP solves for coupled fluid flow and analyte transport in the SAC environment:

- PRESTOMP revises execution periods and time-step control in STOMP input files to minimize the occurrence of STOMP convergence failures.
- PRESTOMP optionally scales the wetted area of one-dimensional vadose zone grids for sites that receive artificial discharges.

#### 33.1.1 Time Stepping Refinement

PRESTOMP will replace the STOMP ~Solution Control card with one that optimizes the time stepping to minimize the chance that the simulation will end in a convergence failure. The algorithm used in PRESTOMP to achieve this was developed recognizing that convergence failures will typically arise when there are large changes in boundary conditions or source terms without reducing the time steps at the times of change. STOMP will automatically reduce the time step four times to achieve convergence. If it fails in those successive attempts, STOMP will quit and issue a convergence failure warning. Commonly, further reduction is all that is needed to achieve a satisfactory solution. To this end, PRESTOMP will start a new execution period for every instance where the recharge rate, or fluid source strength, increases by a factor of maxJump or greater. The user may control the value of maxJump in the PRESTOMP command line, or if not specified by the user, PRESTOMP will assume a value of 1.05 (that is, any boundary condition or source rate increase of at least 5% will cause a new execution period to be requested by PRESTOMP). All new execution periods begin at 1.0 seconds and grow again at the rate already specified in the STOMP input file's ~Solution Control card. (The tilde symbol ~ is used in the STOMP input file format to denote the declaration of a new input card—a related group of inputs).

#### 33.1.2 Vadose Zone Wetted Area Scaling

For the vadose zone transport calculations in one-dimensional grids, an optional AreaX factor can be used to specify a multiplier for the vadose zone area. This is used to represent lateral spreading that would occur where large artificial liquid discharges happen, wherein the actual area of vadose zone transport is larger than the facility footprint given by WIDS (Waste Information Data System) coordinates. When ESP creates input files for STOMP runs, the area derived from the specified coordinates will be multiplied by AreaX; then the x- and y- extents written to the STOMP input file for one-dimensional cases will be the square of the resulting scaled area.

A negative value may be specified for AreaX as input to PRESTOMP to direct the code to compute the AreaX factor for each site and each realization using the  $K_s$ -dependent approach. For this approach, PRESTOMP will compute the effective wetted area of a vadose zone site using the equation

$$A_x = \frac{|Q_{\max}|}{K_{s\min} A_0}, A_x \geq A_{\min}$$

where  $A_X$  = the multiplier of the facility footprint area (dimensionless)  
 $Q_{\max}$  = the maximum artificial liquid discharge rate (m/s)  
 $K_{s|\min}$  = the minimum hydraulic conductivity of all layers for the given site (m/s) and realization  
 $A_0$  = the nominal area (m<sup>2</sup>) defined by the coordinates in the SITE keyword.  
 $A_{\min}$  = minimum area (default 1.0 if not specified by user)

AreaX (variable  $A_X$  is constrained by ESP to be equal to or greater than AreaXmin ( $A_{X|\min}$ ) that may also be optionally entered in the PRESTOMP command line invocation or else will use the default value of 1.0 (area cannot be smaller than that declared for the site in the STOMP input file before PRESTOMP execution). In cases such as those representing tank leaks, it is appropriate to allow  $A_X$  to be less than 1.0 (less than the area of the facility footprint), so in these cases a smaller value of AreaXmin may be entered by the user.

If the PUMP modifier is used, the value of  $Q_{\max}$  is assigned the maximum of the absolute value of the minimum artificial discharge rate (i.e., the largest sink term) and the maximum artificial discharge rate. (This allows a pumping well to be sized in accordance with maximum sink rate rather than maximum recharge).

The horizontal extent of the vadose zone wetted area written by PRESTOMP to the STOMP input (STOMP ~Grid Card) file is

$$x = y = \sqrt{A_X A_0}$$

If the grid of the STOMP “input” file being processed has more than one node in either horizontal dimension, then PRESTOMP will not scale the area.

### 33.1.3 Location in the Processing Sequence

PRESTOMP is designed to be executed after VADER inserts analyte and fluid sources into the STOMP transport input file, and before STOMP is executed to solve for fluid and mass transport. The expected sequence of executions controlled by the ESP code is

1. stompX-pY-1.exe (flow solution)
2. setres-1 (eliminate unneeded files and set restart file)
3. vader-1.exe (calculate releases of analyte mass and fluid volume)
4. prestomp-1.exe (refine time stepping and modify VZ wetted area)
5. stompX-pY-1.exe (mass and fluid transport solution)

In the above sequence “X” refers to the STOMP mode (1 = water; 2 = water+air), while “Y” refers to the processor to which the code is optimized (3 = Pentium III; 4 = Pentium 4 processor).

### 33.1.4 How the Code Is Invoked

PRESTOMP is invoked with the following command-line calling parameters:

```
prestomp-1.exe filename AreaX=val maxJump=val upscale=val  
AreaXmin=val maxStep=val PUMP
```

The command line parameters for PRESTOMP are defined in Table 33.1.

**Table 33.1** Parameters Associated with the Execution of the PRESTOMP Code

Parameter	Data Type	Description
filename	character	Name of the STOMP input file (typically “input”)
AreaX	real	[Optional] Multiplier on vadose zone wetted column area to apply; a negative value will invoke the Ksat-derived method of computing AreaX based on peak artificial discharge rate and minimum Ksat in the profile. If not specified, default value is 1.0.
maxJump	real	[Optional] Threshold of increase in fluid flux at boundary or in a source that will force a new STOMP execution period to avoid convergence failures. If not specified, default value is 1.05.
upscale	real	[Optional] Factor to upscale Ksat-derived AreaX by (used for iterative handling of problematic flow cases by ESP). If not specified, default value is 1.0).
AreaXmin	real	[Optional] Minimum allowed value of AreaX. If not specified, default value is 1.0.
maxStep	real	[Optional] Maximum time step (in years) to allow in any STOMP execution. If not used, defaults are 100.0 yr for STOMP Mode 1 and 1.0 yr for STOMP Mode 2.
PUMP	character	If the modifier PUMP is present in the command line, negative liquid discharges will not be truncated to zero. If not specified, PRESTOMP will by default truncate net negative liquid sources each time step to zero.

Note that all command line arguments except `filename` are optional. The argument `filename` must be the first argument, but all other optional arguments may be entered in any order after `filename` (these are recognized by the modifier). No spaces are allowed within an optional argument. For example,

```
prestomp-1.exe input maxStep=1.0
```

is acceptable, but

```
prestomp-1.exe input maxStep= 1.0
```

is not.

Typically, PRESTOMP is invoked by ESP. In this case, the inputs are drawn from the ESD file as follows:

```
filename   : ESP assumes “input”
AreaX      : from ESD file SITE keyword, modifier AreaX=(value)
maxJump    : ESP assumes 1.01
```

upScale : ESP starts with 1.0, and increases as needed (see below)  
AreaXmin : from ESD file SITE keyword, modifier AreaXmin=(value)  
maxStep : provided by user input in the “stochastic\_stomp.key” file  
PUMP : from ESD file SITE keyword, modifier PUMP

The ESP code is designed to iteratively solve a vadose zone flow and transport problem with successively larger vadose zone wetted areas if convergence is not reached in a pre-specified number of time steps. The limit on the number of time steps is controlled in the vadose zone STOMP template (50,000 is typical in current SAC templates). If this limit is reached, ESP will multiply the maxJump parameter last used by 1.07 and attempt to solve the vadose zone flow and transport case. This approach is needed because occasionally the minimum Ksat / maximum discharge method for selecting the vadose zone wetted area (see above) results in a set of vadose zone properties that are physically unable to permit the necessary fluid discharge that records indicate happened. In these cases, ESP can iteratively upscale the vadose zone wetted area by factor upScale until a solution is reached or a hard limit is reached. (The hard limit is that the area of the vadose zone site cannot exceed the area of the Hanford Site; when this hard limit is reached, it is flagged as an error that clearly indicates something else is wrong and requires user attention).

## 33.2 File Definitions

The PRESTOMP code reads one input file and writes one or more output files.

### 33.2.1 Input Files

PRESTOMP reads a STOMP input file, usually named “input”. An example STOMP input file is provided in Table 33.2. Note that PRESTOMP leaves comments in the STOMP “input” file to indicate the modifications made as follows;

- In the ~Solution Control Card, PRESTOMP inserts one comment line to indicate that it tuned the execution periods (to avoid convergence failures) and when this modification was made.
- In the ~Grid Card, PRESTOMP inserts two comment lines indicating that it modified the card and when the modification was made, and noting the AreaX factor applied in changing the horizontal extent of the domain.
- In the ~Source Card, PRESTOMP inserts one comment line after the declaration of any density volumetric source to indicate that it scaled the source to match changes made to the horizontal extent of the domain.

**Table 33.2** Example STOMP Input and Output File “input” to the PRESTOMP Code

```
#SAC STOMP input created by ESP    06/12/2006 - 10:27:37
#SAC Case ID      : Action Assessment using a limited suite of sites
#SAC Template     : vadose/218-W-11/C14/template_stomp2.key
#SAC Site ID      : 218-W-11
#SAC Site Points  : 4
#SAC Site Eastings (m) : 5.6615675E+05, 5.6625312E+05, 5.6625312E+05,
5.6615675E+05,
#SAC Site Northings (m): 1.3627042E+05, 1.3627042E+05, 1.3636680E+05,
1.3636680E+05,
#SAC Site Area (m^2)  : 9.2876094E+03
#SAC Analyte       : C14
#SAC Realization    : 1

~Simulation Title Card
1,
216T-4/2,
W E Nichols,
Pacific Northwest National Laboratory,
January 28 2006,
7 PM PST,
3,
Template 216T - For shallow disposal sites (e.g. Cribs, Burial Grounds)
North 200 West Area (T Areas), Cribs/Burial Grounds, Waste Chemistry Designation
4
Stratigraphy Data Source: "Templates T_NRV_2006-01-25_DCR-0038.xls" in EDTP
vadose_2006-01-26_DCR-0038

~Solution Control
# Execution periods tuned by PRESTOMP: 06/12/2006 14:47:24
Restart,
Water-Air Transport Courant,
8,
0.,s,504921600.,s,1.0,s,1.0,yr,1.5,8,1.0E-6,
504921600.,s,2493072000.,s,1.0,s,1.0,yr,1.5,8,1.0E-6,
...
34081516800.,s,319261824000.,s,1.0,s,1.0,yr,1.5,8,1.0E-6,
50000,
Zero,
Zero,
0,

~Grid Card
# AreaX modified by PRESTOMP: 06/12/2006 14:47:24
# AreaX      =      1.00
Cartesian,
1,1,650,
0.00000E+00,m, 9.63766E+01,m,
0.00000E+00,m, 9.63766E+01,m,
407.0,ft,206@0.5,ft,190@0.2,ft,254@0.5,ft,

~Rock/Soil Zonation Card
7,
B(4H),1,1,1,1,617,650,
Hg_2W(4H),1,1,1,1,467,616,
Hgs_2W(4I1),1,1,1,1,397,466,
```

```

PPlz(4I1),1,1,1,1,347,396,
PPlc(4I1),1,1,1,1,257,346,
PPlz(4I1),1,1,1,1,207,256,
Rg_2W(4I2),1,1,1,1,1,206,

#SAC VADER Source Range,1,1,1,1,617,650,

~Mechanical Properties Card
B(4H),2.65000E+03,kg/m^3,2.70110E-01,2.70110E-01,,millington and quirk,
Hg_2W(4H),2.65000E+03,kg/m^3,1.51580E-01,1.51580E-01,,millington and quirk,
Hgs_2W(4I1),2.65000E+03,kg/m^3,2.72800E-01,2.72800E-01,,millington and quirk,
PPlz(4I1),2.66600E+03,kg/m^3,4.15580E-01,4.15580E-01,,millington and quirk,
PPlc(4I1),2.66700E+03,kg/m^3,2.89140E-01,2.89140E-01,,millington and quirk,
Rg_2W(4I2),2.65000E+03,kg/m^3,2.80510E-01,2.80510E-01,,millington and quirk,

~Hydraulic Properties Card
B(4H),9.14390E-04,hc cm/s,9.14390E-04,hc cm/s,9.14390E-04,hc cm/s,
Hg_2W(4H),9.17470E-04,hc cm/s,9.17470E-04,hc cm/s,9.17470E-04,hc cm/s,
Hgs_2W(4I1),5.45270E-05,hc cm/s,5.45270E-05,hc cm/s,5.45270E-05,hc cm/s,
PPlz(4I1),8.10580E-05,hc cm/s,8.10580E-05,hc cm/s,8.10580E-05,hc cm/s,
PPlc(4I1),7.86220E-04,hc cm/s,7.86220E-04,hc cm/s,7.86220E-04,hc cm/s,
Rg_2W(4I2),1.51660E-04,hc cm/s,1.51660E-04,hc cm/s,1.51660E-04,hc cm/s,

~Saturation Function Card
B(4H),Nonhysteretic van Genuchten,1.85260E-02,1/cm,1.41560E+00,1.05050E-01,,
Hg_2W(4H),Nonhysteretic van Genuchten,1.61570E-02,1/cm,1.76810E+00,1.40750E-01,,
Hgs_2W(4I1),Nonhysteretic van Genuchten,7.89910E-03,1/cm,2.22300E+00,1.33400E-
01,,
PPlz(4I1),Nonhysteretic van Genuchten,4.89450E-03,1/cm,2.22150E+00,9.63280E-02,,
PPlc(4I1),Nonhysteretic van Genuchten,1.17900E-02,1/cm,1.77390E+00,1.85330E-01,,
Rg_2W(4I2),Nonhysteretic van Genuchten,1.42810E-02,1/cm,1.66970E+00,1.31310E-01,,

~Aqueous Relative Permeability Card
B(4H),Mualem,,
Hg_2W(4H),Mualem,,
Hgs_2W(4I1),Mualem,,
PPlz(4I1),Mualem,,
PPlc(4I1),Mualem,,
Rg_2W(4I2),Mualem,,

~Gas Relative Permeability Card
B(4H),Mualem,,
Hg_2W(4H),Mualem,,
Hgs_2W(4I1),Mualem,,
PPlz(4I1),Mualem,,
PPlc(4I1),Mualem,,
Rg_2W(4I2),Mualem,,

~Solute/Fluid Interaction Card
1,
C14,1.05000E-
05,cm^2/s,0.00000E+00,cm^2/s,Constant,1.63516E+00,m^3/m^3,,5.71500E+03,yr,
0,
0,
0,

```

```

~Solute/Porous Media Interaction Card
B(4H),0.09,m,,,
C14,0.00000E+00,mL/g,1.0,
Hg_2W(4H),0.09,m,,,
C14,0.00000E+00,mL/g,1.0,
Hgs_2W(4I1),0.088,m,,,
C14,0.00000E+00,mL/g,1.0,
PPlz(4I1),0.031,m,,,
C14,0.00000E+00,mL/g,1.0,
PPlc(4I1),0.031,m,,,
C14,0.00000E+00,mL/g,1.0,
Rg_2W(4I2),0.09,m,,,
C14,0.00000E+00,mL/g,1.0,

~Initial Conditions Card
Gas Pressure,Aqueous Saturation,
7,
Gas Pressure,101325.0,Pa,0.0,1/m,0.0,1/m,0.0,1/m,1,1,1,1,1,650,
Aqueous Saturation ROCK,0.2637,,B(4H),
Aqueous Saturation ROCK,0.2230,,Hg_2W(4H),
Aqueous Saturation ROCK,0.2118,,Hgs_2W(4I1),
Aqueous Saturation ROCK,0.2178,,PPlz(4I1),
Aqueous Saturation ROCK,0.2816,,PPlc(4I1),
Aqueous Saturation ROCK,0.2259,,Rg_2W(4I2),

~Boundary Conditions Card
2,
Top,Neumann Aqueous,Dirichlet Gas,Volumetric Concentration Solute,
1,1,1,1,650,650,16,
0.000000000000E+00,s,-4.00000E+00,mm/yr,1.0,101325.0,Pa,1.0,,,
5.049216000000E+08,s,-4.00000E+00,mm/yr,1.0,101325.0,Pa,1.0,,,
...
3.192302016000E+11,s,-1.00000E+00,mm/yr,1.0,101325.0,Pa,1.0,,,
Bottom,Dirichlet Aqueous,Neumann Gas,Zero Flux Solute,
1,1,1,1,1,1,41,
0.0,yr,101325.0,Pa,0.0,0.0,m/s,0.0,,,
1.577880000000E+07,s,1.49166E+05,Pa,1.0,0.0,m/s,1.0,,,
...
5.743483200000E+09,s,1.49166E+05,Pa,1.0,0.0,m/s,1.0,,,
1.0E+6,yr,101325.0,Pa,0.0,0.0,m/s,0.0,,,

~Output Control Card
1,
1,1,617,
1,1,yr,m,6,6,6,
6,
aqueous pressure,pa,
aqueous saturation,null,
z aqueous vol,mm/yr,
aqueous courant number,null,
solute volumetric conc.,C14,1/m^3,
solute source integral,C14,null,
12,
1.925078400000E+09,s,
...
3.189146688000E+11,s,

```

```
2,  
aqueous saturation,null,  
solute volumetric conc.,C14,1/m^3,  
  
~SAC Release Plane Card  
40,  
0.0,s,1,  
1.577880000000E+07,s,33,  
3.850027200000E+09,s,40,  
...  
5.743483200000E+09,s,33,  
  
~Balance Card  
12,  
1.925078400000E+09,s,  
..  
3.189146688000E+11,s,  
  
~SAC Remediation Card  
0,  
  
~Source Card  
1,  
# Additional source term added by VADER: 06/12/2006 - 14:47:24  
# VADER volSrc = 4.81246E+04 scaleFactor = 1.00000E+00  
Solute Density,C14,1,1,1,1,617,650,37,  
# Density source scaled to new volume resulting from AreaX by PRESTOMP:  
06/12/2006 14:47:24  
0.0000000E+00,s,0.0000000E+00,1/s m^3,  
0.0000000E+00,s,0.0000000E+00,1/s m^3,  
3.1536000E+07,s,0.0000000E+00,1/s m^3,  
...  
5.3645760E+08,s,0.0000000E+00,1/s m^3,  
# CFEST Template      : cfest/H3-template/cfest.lpl  
# Nearest CFEST Node: 796  
  
#END
```

### 33.2.2 Output Files

PRESTOMP echoes principal control data such as run date, analysis title, realization, site, analyte, release models, and coefficient values to a log file. PRESTOMP will record unusual events and error messages in this file.

If PRESTOMP completes its analysis without any problems, it will modify the STOMP input file (“input”) to reflect the analysis and then create a “prestomp.done” file. However, if PRESTOMP fails for any reason, it will create a “prestomp.fail” file. These files are used by ESP to make sure that all of the modeling is error free or to log any errors that occur.



## 34.0 RELUP – STOMP Release Time Step Threshold Reprocessor

### 34.1 Overview

RELUP is a post-processor utility code for STOMP used either by ESP or independently. RELUP provides the means to restrict the minimum time duration for a release record in the “release” file written by STOMP in the SAC framework.

#### 34.1.1 Location in the Processing Sequence

RELUP is designed to be executed after STOMP runs and produces a “release” file. The expected sequence of executions controlled by the ESP code is

1. stompX-pY-1.exe (flow solution)
2. setres-1 (eliminate unneeded files and set restart file)
3. vader-1.exe (calculate releases of analyte mass and fluid volume)
4. prestomp-1.exe (refine time stepping and modify VZ wetted area)
5. stompX-pY-1.exe (mass and fluid transport solution)
6. relup-1.exe

In the above sequence “X” refers to the STOMP mode (1=water, 2=water+air), while “Y” refers to the processor to which the code is optimized (3 = Pentium III, 4 = Pentium 4 processor).

#### 34.1.2 How the Code Is Invoked

RELUP is invoked with the following command line calling parameters:

```
relup-1.exe filename threshold
```

where “filename” is the name of the STOMP release file, and “threshold” is the minimum time threshold to impose on the release records in `filename`. The units are implicitly the same as the time units used in the release records in `filename` (usually seconds).

Note that both command line arguments are required. If the parameter used for `filename` contains “help”, then the code will display information about the code and terminate without running.

Typically, RELUP is invoked by ESP. In this case, the inputs are drawn from the ESD file as follows:

```
filename    :   ESP assumes “release”  
threshold   :   One day (86,400 seconds)
```

### 34.2 File Definitions

RELUP reads one input file and writes one output file.

### 34.2.1 Input Files

RELUP reads a STOMP release file, usually named “release”.

### 34.2.2 Output Files

RELUP writes an output STOMP release file. RELUP renames the file specified in `filename` to `filename0`, that is, the same name with a zero character appended to preserve the original release file written by STOMP. For example, “release” is renamed “release0”. Then RELUP opens a new file with the name specified in `filename`. RELUP then reads each line in `filename0` and does the following:

- If the first character of a line is “#” or “~” (comments and title indicators), the line is copied directly to `filename0`.
- If the line is the first release record, it is copied directly to `filename0`.
- If the line is a subsequent release record, the time duration of the record is checked to ensure it is equal to or greater than `threshold`. If it is, then the record is written to `filename0`. If it is not, then the quantity released is accumulated to the next record until the time duration of multiple release records exceeds `threshold`.
- Any accumulated mass by the last release record is recorded regardless of the time step duration.

## 35.0 RLGRAB – Extractions from the Release Module

### 35.1 Overview

RLGRAB is a tool for retrieving release and inventory time-profile data from the VADER runs as written to the “vader.table” files. RLGRAB can also retrieve release model parameters used in the SAC runs. RLGRAB retrieves data for one analyte per run.

#### 35.1.1 Location in the Processing Sequence

RLGRAB retrieves data from “vader.table” files built by the Release Module (VADER) code. RLGRAB can be run anytime after all the VADER runs in a SAC simulation are completed.

#### 35.1.2 How the Code Is Invoked

RLGRAB only runs under the Linux operating system. RLGRAB is executed through the following Bourne Shell or C Shell command:

```
rlgrab.exe "RLGRABKeyFileName"
```

For this command, “rlgrab.exe” is the name of the executable program, and “RLGRABKeyFileName” is the name of the RLGRAB control keyword file. If the RLGRAB control keyword file name is not supplied, it defaults to a file in the current directory named “rlgrab.key”. The executable program name and the keyword file name can include path information.

#### 35.1.3 Memory Requirements

The RLGRAB code uses dynamic memory allocation. It is expected that all of the runs of the RLGRAB code will require less than 24 MB of memory.

### 35.2 File Definitions

The RLGRAB code reads one or more input files and writes one or more output files.

#### 35.2.1 Input Files

RLGRAB reads a control keyword file, an optional site name list, and one or more VADER table files. An example keyword file is provided in Table 35.1. The format of the table files is discussed by Eslinger et al. (2006a, Section 5.5.2) and is not discussed further here.

**Table 35.1** Example Keyword File for RLGRAB

TITLE "Test Decaying and Filling In"
SITE "241-A-101" "241-A-102"
ANALYTE "Sr90" HALFLIFE=28.78 DECAYYR=1944
WASTETYPE "CMNT" "SOIL"
YEARS START=1944 END=22050
VARIABLE "RMD_IN" "RMD_OUT" "QTY_IN"
PATHNAME "/home/ANALYSIS4/CA1_median/vadose/"
REALIZATION 1 MAXREAL=1

END
-----

RLGRAB attempts to open and read all “vader.table” files in the subdirectories below the “vadose” subdirectory which match the analyte, site names, and realizations provided in the “rlgrab.key” file. If a “vader.table” file cannot be opened or read, RLGRAB notes that occurrence in the RLGRAB.Con file and proceeds. Therefore, the user must verify that all “vader.table” files have been successfully processed before proceeding to post-retrieval analysis. To verify that all files were actually processed, inspect the “Number failed to open” line near the bottom of the “RLGRAB.Rpt” file.

### 35.2.2 Output Files

RLGRAB writes one or two output files. The report file is written for every run of the code and has a default name of “RLGRAB.Rpt”. This text file documents the processing actions and contains any error messages. The other output file has a default name of “RLGRAB.Con” and contains yearly consolidated release quantities. The OUTFILE keyword can be used to provide alternative file names for these two files. The “RLGRAB.Con” file is a text file containing a few header lines and a data matrix. The data matrix is often imported into a plotting package, statistical package, or spreadsheet for further processing and display.

## 35.3 Keyword Definitions for RLGRAB

In general, the keywords for the RLGRAB code can be entered in any order. The following restrictions apply to keyword order:

- Only one of the PERIOD or YEARS keywords should be used.
- The END keyword must be the last keyword in the file.

### 35.3.1 ANALYTE Keyword for RLGRAB

The ANALYTE keyword is used to define the analytes for which release data will be extracted. The following is this keyword’s syntax:

```
ANALYTE ["quote 1"] {HALFLIFE=n1} {DECAYR=N2}
```

Analytes chosen must be a subset of the analytes used in the fate and transport activities controlled through the ESD keyword file. Only a single analyte keyword may be entered. The optional numerical data value associated with the optional modifier HALFLIFE defines the half-life (years) of the analyte. A value of 1.0E+34 is used if HALFLIFE is not used. The optional numerical value associated with the optional DECAYR is a calendar year to which all radioactive data will be decay corrected. If this value is not defined, the data are not decay corrected (and thus represent the year of disposal). An example keyword is the following:

```
ANALYTE "Sr90" HALFLIFE=28.78 DECAYR=1944
```

### 35.3.2 END Keyword for RLGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword’s syntax:

END

### 35.3.3 OUTFILE Keyword for RLGRAB

The optional OUTFILE keyword is used to modify the default output file names. The following is this keyword's syntax:

```
OUTFILE [RAW="quote 1"]
```

The quote string associated with the RAW modifier contains the path name and base file name for output. The report file will have the location and name derived from appending “.Rpt” to the quote string. The output results file will have the location and name derived from appending “.Con” to the quote string. An example use of this keyword is the following:

```
OUTFILE "/home/ANALYSIS4/CA1_median/Case2"
```

This keyword would result in naming the two output files as follows:

```
"/home/ANALYSIS4/CA1_median/Case2.Rpt"  
"/home/ANALYSIS4/CA1_median/Case2.Con"
```

### 35.3.4 PATHNAME Keyword for RLGRAB

The optional PATHNAME keyword is used to identify path name to the vadose directory of the analysis being examined. The following is this keyword's syntax:

```
PATHNAME ["quote"]
```

The quote string containing the pathname must end with the “/” character. The path associated with the “RLGRAB.key” file will be used if this keyword is not entered. An example use of this keyword is the following:

```
PATHNAME "/home/ANALYSIS4/CA1_median/vadose/"
```

### 35.3.5 PERIOD Keyword for RLGRAB

The PERIOD keyword identifies the start and stop years for the analysis being examined. The following is this keyword's syntax:

```
PERIOD [START=N1] {END=N2} [STOP=N3]
```

The PERIOD and YEARS keywords identify the same data. Only one of these two keywords should be used in a single run of the code. The modifier START and the associated value N1 identify the year to start extracting data. The modifier STOP and the associated value N3 identify the year to stop extracting data. The optional modifier END and the associated value N2 also identify the year to stop extracting data. The STOP modifier takes precedence if both the END and STOP modifiers are used. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example PERIOD keyword that extracts data from 1944 through 3050:

```
PERIOD START=1944 STOP=3050
```

### 35.3.6 REALIZAT Keyword for RLGRAB

The REALIZAT (or REALIZATION) keyword identifies the realization numbers to process and the total number of realizations that were simulated. The following is this keyword's syntax:

```
REALIZAT {MAXREAL=N1} [ N2-N3 | N4 {N5} ... {N6} | ALL]
```

The total number of realizations processed in the analysis is identified by the modifier MAXREAL. A default value of 100 realizations will be used if the modifier MAXREAL is not present. The specific realizations to process can be specified in three different ways. First, a range of realizations (small to large) can be entered by providing two numbers separated by a hyphen. Second, a list of specific realizations can be specified as individual realization numbers separated by spaces. Finally, all realizations can be specified by entering the modifier ALL. The following example REALIZAT keywords all request realizations 1 through 6 of a data set containing exactly six realizations.

```
REALIZAT ALL MAXREAL=6  
REALIZAT 1 2 3 6 5 4 MAXREAL=6  
REALIZAT 1-6 MAXREAL=6
```

### 35.3.7 SITE Keyword for RLGRAB

The optional SITE keyword is used to identify a list of site IDs to process. The following is this keyword's syntax:

```
SITE ["quote1"] ("quote2") ... {"quoteN"}
```

The quote strings entered on this keyword must each be a valid site ID for the analysis being examined. More than one SITE keyword can be entered. The effect of any SITE keywords and the SITEFILE keyword is cumulative. An example use of this keyword is the following:

```
SITE "200_ETF" "200-E-100" "200-E-102" "200-E-103"
```

### 35.3.8 SITEFILE Keyword for RLGRAB

The optional SITEFILE keyword is used to identify an input file that contains a list of site IDs to process. The following is this keyword's syntax:

```
SITEFILE ["quote"]
```

The effect of any SITE keywords and the SITEFILE keyword is cumulative. An example use of this keyword is the following:

```
SITEFILE "SiteID_Case10.dat"
```

### 35.3.9 TITLE Keyword for RLGRAB

The optional TITLE keyword is used to define a single-line problem title that will be written to output files for labeling purposes. The following is this keyword's syntax:

```
TITLE ["quote"]
```

An example use of this keyword is the following:

```
TITLE "SAC Special Run - Retrieval 17"
```

### 35.3.10 VARIABLE Keyword for RLGRAB

The VARIABLE keyword is used to identify the type of output to process. The following is this keyword's syntax:

```
VARIABLE [ "quote1" ] ( "quote2" } ... { "quoteN" )
```

This keyword specifies the variables in the “vader.table” files to extract and sum. Entries in the quote strings are not case sensitive for this keyword. More than one VARIABLE keyword can be entered. The effect of multiple VARIABLE keywords is cumulative. The user can retrieve any variable in the “vader.table” file heading by name. Example entries for this keyword are the following:

```
VARIABLE "Qty_In"      ! Extract the annual inventory deposit
VARIABLE "RMD_IN"      ! Extract the annual remediation import
VARIABLE "RMD_OUT"     ! Extract the annual remediation export
VARIABLE "Release"     ! Extract the annual quantity released
VARIABLE "CumQty"      ! Extract the net inventory remaining
VARIABLE "CumRel"      ! Extract the cumulative quantity released
VARIABLE "Qw(cm)"      ! Extract the annual infiltration rate
```

Normally RLGRAB should be used to retrieve only one time-dependent variable per run, since it makes no sense for most of the variables to be added together. However, retrievals that combine inventory deposits and remediation transfers, in order to gain a picture of total inventory at a site not accounting for release (decayed to deposit or transfer year), can be performed. This is accomplished by setting the VARIABLE keyword arguments to more than one variable, as in:

```
VARIABLE "QTY_IN" "RMD_IN" "RMD_OUT" ! Retrieve all deposits and transfers
VARIABLE "RMD_IN" "RMD_OUT"           ! Focus on remedial actions
VARIABLE "QTY_IN" "RMD_IN"            ! Focus on deposits and imports
VARIABLE "QTY_IN" "RMD_OUT"           ! Focus on deposits and exports
```

### 35.3.11 WASTETYP Keyword for RLGRAB

The WASTETYP keyword is used to identify a list of waste types to process. The following is this keyword's syntax:

```
WASTETYP [ "quote1" ] ( "quote2" } ... { "quoteN" )
```

This keyword specifies the waste types in the “vader.table” files to extract and sum. Entries in the quote strings are not case sensitive for this keyword. The text in the quote string defining a waste type needs to actually appear in the “vader.table” file “Waste” heading group in order for any data to be extracted. Matching is declared whenever the quote string on this keyword contains the specified waste type as an embedded string (for example, “soilsn” is retrieved as “SOIL”). More than one WASTETYP keyword can be entered. The effect of multiple WASTETYP keywords is cumulative.

To obtain results organized by waste type, it is necessary to perform a separate retrieval for each waste type. One approach is to run an initial RLGRAB retrieval to build a "RLGRAB.Rpt" file. The report file contains information on all waste forms output to the "vader.table" file for all sites, whether or not they were actually added to the totals in the "RLGRAB.Con" file. These appear as the "Specs" records in the report file.

Specify the string "AGGREGATE" for all releases from solid waste types in the vadose zone. If "AGGREGATE" is specified, do not specify any other solid waste types. Specify the string "LIQUID" for all releases from liquid waste streams to the vadose zone. A run extracting all liquid releases to both the vadose zone and the river should specify "LIQUID" and "RIVER". An example use of this keyword to extract releases from soil waste types is the following:

```
WASTETYPE "SOIL"
```

This example keyword will extract and sum data for the following waste types: "soil", "soil1f", "soil1n", "soil3f", "soil3n", "soils", "soilsf", "soilsn", "soilss", "soilst", and "soiltu".

### 35.3.12 YEARS Keyword for RLGRAB

The YEARS keyword identifies the start and stop years for the analysis being examined. The following is this keyword's syntax:

```
YEARS [START=N1] [STOP=N2]
```

The PERIOD and YEARS keywords identify the same data. Only one of these two keywords should be used in a single run of the code. The modifier START and the associated value N1 identify the year to start extracting data. The modifier STOP and the associated value N2 identify the year to stop extracting data. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example YEARS keyword that extracts data from 1944 through 3050:

```
YEARS START=1944 STOP=3050
```



## 36.0 SETRES – STOMP Restart File Handling

### 36.1 Overview

SETRES is a small processing utility code used to rename files following a STOMP flow simulation to prepare for a STOMP transport simulation in the same directory. Its purpose is to gather the names of all restart files written by STOMP during the flow solution, identify the last one that provides the conditions that will be used as initial conditions for the transport solution, and rename the final file to the standard “restart” name while deleting all other restart files.

#### 36.1.1 Location in the Processing Sequence

SETRES is designed to be executed after a successful STOMP flow solution. The expected sequence of executions controlled by the ESP code is:

1. stompX-pY-1.exe (flow solution)
2. setres-1.exe (eliminate unneeded files and set restart file)

In the above sequence “X” refers to the STOMP mode (1=water, 2=water+air), while “Y” refers to the processor to which the code is optimized (3 = Pentium III, 4 = Pentium 4 processor).

#### 36.1.2 Memory Requirements

The SETRES code uses dynamic memory allocation. It is expected that all of the runs of the SETRES code will require less than 2 MB of memory.

#### 36.1.3 How the Code Is Invoked

SETRES only runs under the Linux operating system. SETRES is executed through the following Bourne Shell or C Shell command:

```
setres-1.exe
```

For this command, “setres-1.exe” is the name of the executable program. The executable program name can include path information.

### 36.2 File Definitions

SETRES gathers a list of all files in the active directory named “restart.nnn” where .nnn is any integer number. STOMP can write multiple restart files during execution, appending the time step number as the file name extension. The final restart file at the conclusion of the flow simulation contains the state variable values desired for use as initial conditions in the transport solution. When operating in RESTART mode, STOMP expects these values to be provided in a file simply named “restart”. SETRES identifies the “restart.nnn” file with the highest integer file name extension, renames that file “restart” and deletes the other “restart.nnn” files to avoid confusion with the “restart.nnn” files that will be written during the subsequent transport simulation.



## **37.0 SIMS – SIM Inventory Data Processing**

### **37.1 Overview**

The SIM (Soil Inventory Model) information is received in large Excel files that have multiple worksheets. These worksheets contain stochastic volume and release amounts for many liquid release sites at Hanford. The SIMS utility code is used to reformat and decay correct the data so they can be imported into the SAC inventory database. SIM data, as received, are decay corrected to a single common year. The SIMS processor back-decays data, as necessary, to correct activity amounts (Ci) for radioactive analytes to the year of disposal.

#### **37.1.1 Location in the Processing Sequence**

The SIMS utility code is a preprocessor to the inventory database. It is used before any other SAC code.

#### **37.1.2 Memory Requirements**

The SIMS code uses dynamic memory allocation. It is expected that all of the runs of the SIMS code will require less than 2 MB of memory.

#### **37.1.3 How the Code Is Invoked**

SIMS runs under the Windows operating system in a DOS box. A run of SIMS is initiated by entering the following command line:

```
SIMS FilePrefix
```

For this command, “SIMS.EXE” is the name of the executable program, and “FilePrefix” is the prefix for an input file name. The SIMS code attempts to open an input file named “FilePrefix.csv”. If SIMS cannot find the specified file, then the code will terminate execution after writing an error message to the standard output device.

### **37.2 File Definitions**

The SIMS code reads two input files and writes nine output files.

#### **37.2.1 Input Files**

The first input file for the SIMS code contains the data for a single release area (see the processing description in Section 37.3). This file name always ends in “.csv”. This data file has the following structure:

- Line 1: This line contains heading information including the SIM data set name, the processing date and time, and labels for volume and concentration distributions. This line is skipped during processing.
- Line 2: This line contains column headings, including the site, year, analyte, units, and statistical labels. This line is also skipped during processing.

Lines 3 on: Each following line contains 78 data values. These values are a data index, an analyte index, the site ID, the year of processing, the analyte ID, the data units, and statistical quantities for three data sets (total quantity, total volume, and concentration). The total quantity information is skipped, and the volume and concentration distributions are used instead.

The first 4 lines from a SIMS input data file are shown in Table 37.1. The lines are wrapped in the table, so a blank line is used to indicate the beginning of the next data line. A production data set may contain 10,000 or more lines of data.

The second input file for SIMS is a library of radioactive chain decay information that is always named “RMDLIB.DAT”. The format of this file is not described further because the user typically does not modify this file.

### 37.2.2 Output Files

The SIMS code writes a text log file for every run of the code. This file is named “SIMS.OUT”. The code also writes a text error-message file named “L.ERR”. These files should always be examined to determine whether unexpected errors were encountered during the run of the code.

The SIMS code writes a primary output file and a number of secondary files that are not used further in the data processing. If the input file is named “CASE.csv”, then the primary output file is named “CASE.out”. Thus, the input “\*.csv” file and the output file share the same file name prefix. The primary output file is then imported into the inventory database.

Excerpts from the primary output file from SIMS is provided in Table 37.2. The first line shows as wrapped to the second line in this example. This line contains variable names for use in the inventory database. These data represent the volume and quantities of each analyte in each waste stream as a statistical distribution made up of 23 points defined as a cumulative distribution function.

A number of additional files are written by the SIMS code. The data file named “STREAMS.OUT” should be ignored. If the input file is named “CASE.csv”, then the following additional files are written:

- CASE.qaf – a file that contains data for quality assurance checks on mean values
- CASE.qaM – a file that contains data for quality assurance checks on median values
- CASE.005 – a file that contains data for quality assurance checks on 0.5% values
- CASE.05 – a file that contains data for quality assurance checks on 5% values
- CASE.95 – a file that contains data for quality assurance checks on 95% values
- CASE.995 – a file that contains data for quality assurance checks on 99.5 values.

These files should also be ignored.



```
sim_7_Jun_04, 216-A-1, liquid,1955, V0900 , volume, 1.0384E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0950 , volume, 1.0513E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0995 , volume, 1.0726E+02
sim_7_Jun_04, 216-A-1, liquid,1955, Mean , U234, 4.2170E-04
sim_7_Jun_04, 216-A-1, liquid,1955, SD , U234, 3.7018E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0005 , U234, 2.5605E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0050 , U234, 6.8501E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0100 , U234, 9.7731E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0150 , U234, 1.2356E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0200 , U234, 1.4664E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0250 , U234, 1.7029E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0300 , U234, 1.9533E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0350 , U234, 2.2036E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0400 , U234, 2.4744E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0450 , U234, 2.7727E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0500 , U234, 3.1079E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0550 , U234, 3.4545E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0600 , U234, 3.8541E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0650 , U234, 4.3020E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0700 , U234, 4.8320E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0750 , U234, 5.4677E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0800 , U234, 6.2969E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0850 , U234, 7.3950E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0900 , U234, 8.9096E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0950 , U234, 1.1597E-03
sim_7_Jun_04, 216-A-1, liquid,1955, V0995 , U234, 2.1192E-03
sim_7_Jun_04, 216-A-1, liquid,1955, Mean , U235, 1.8645E-05
```

### 37.3 Procedure for Processing SIM files for Input to Inventory Database

The SIM data are received in large Excel files that have multiple sheets. Each file has a sheet titled “Legend” and additional sheets with release data for specific areas. Each of these sheets are saved to a comma-separated variable (.csv) file and processed with the SIMS program. The following steps are required in preparing the “.csv” files:

1. Open a SIM Excel file.
2. Open a sheet in the file for a specific waste area.
3. In cell A1, change the contents (initially “Total Inventory”) to the set name to be assigned to this SIM case. Typically the set name is SUM\_date (SIM\_07\_Jun\_2004 for example).
4. Save the sheet as a “.csv” file with the area name and extension “.csv”. For example, the Excel sheet named “Solid Waste Zone” is given the file name of “SOLIDWASTE.csv”.
5. Repeat this process until data for all waste areas have been saved to disk as “.csv” files.

The next step is to run the SIMS program to convert each of the “.csv” files into data files that can be imported into the inventory database. If there were four input files named “BPLANT.CSV”, “BFARM.CSV”, “200WPONDS.CSV”, and “200EPONDS.CSV”, then the following four entries would be made at a DOS prompt (SIMS only runs under the Windows operating system):

```
SIMS BPLANT
```

```
SIMS BFARM
SIMS 200WPONDS
SIMS 200EPONDS
```

This sequence of four runs results in the four output files named “BPLANT.OUT”, “BFARM.OUT”, “200WPONDS.OUT”, and “200EPONDS.OUT”. The “\*.OUT” files are then input to the inventory database. The database import would be easier if the extension on each of these files were “.csv” rather than “.out”. Copy the “.out” files to another directory before changing the extension to “.csv” so the input data for SIMS are not destroyed.





## **38.0 STOCHASTIC – Stochastic Values Generation**

### **38.1 Overview**

This utility is a standalone code that generates values defined by statistical distributions. Output options include individual generated values and summary statistics. It can be used to check the effects of stochastic variables before they are included in a SAC simulation.

#### **38.1.1 Location in the Processing Sequence**

The STOCHASTIC code can be executed independent of other codes in the SAC processing sequence. However, it can also be coupled with other utility codes (such as INV\_MED, STO\_MED, AND IMP\_MED) to generate keyword files populated with the median values for each of the stochastic variables.

#### **38.1.2 How the Code Is Invoked**

STOCHASTIC can run under either the Windows or the Linux operating systems. Under the Windows operating system (Releases 2000 or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC "Keyfilename"
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe "Keyfilename"
```

For these commands, “STOCHASTIC.EXE” or “stochastic.exe” is the name of the executable program, and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If STOCHASTIC is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If STOCHASTIC cannot find or open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **38.1.3 Memory Requirements**

The STOCHASTIC code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. A reasonably large example run—where the STOCHASTIC code used six stochastic variables and one million realizations—required 16 MB of memory (on a Windows XP machine).

### **38.2 File Definitions**

The STOCHASTIC code reads a single input file and writes up to three output files.

### 38.2.1 Input Files

The single input file for the STOCHASTIC code is a control keyword file. An example keyword file is provided in Table 38.1. Detailed descriptions for the individual keywords are provided in Section 38.3.

**Table 38.1** Example Keyword File for STOCHASTIC

```
REPORT "Step1.Rpt"
USER "John Q. Doe"
TITLE "Generate Median Values for Vadose Zone Stochastic Data"
SEED 12456564.0
REALIZAT 1000001
SINGLEKEY PERCENT=0.50 FILE="Step1.Val"
!
STOCHASTIC "30" "X" 1 1.000
STOCHASTIC "31" "X" 10 20
    0.00000E+00 1.000000E+00
    5.00000E-01 1.000000E+00
    5.54012E-01 1.044167E+00
    6.04938E-01 1.088333E+00
    6.52778E-01 1.132500E+00
    6.97531E-01 1.176667E+00
    7.39198E-01 1.220833E+00
    7.77778E-01 1.265000E+00
    8.13272E-01 1.309167E+00
    8.45679E-01 1.353333E+00
    8.75000E-01 1.397500E+00
    9.01235E-01 1.441667E+00
    9.24383E-01 1.485833E+00
    9.44444E-01 1.530000E+00
    9.61420E-01 1.574167E+00
    9.75309E-01 1.618333E+00
    9.86111E-01 1.662500E+00
    9.93827E-01 1.706667E+00
    9.98457E-01 1.750833E+00
    1.00000E+00 1.795000E+00
STOCHASTIC "52" "X" 6 1.000 2.590 16.900
!
END
```

### 38.2.2 Output Files

The STOCHASTIC code writes from one to three output files:

- **Report File.** The report file is an ASCII file containing information about the run of the STOCHASTIC program. It is written for every run of the code. All error messages (except for those indicating the report file could not be opened) are written to this file.
- **SINGLEKEY File.** A revised STOCHASTIC keyword can optionally be written to an output file for every input STOCHASTIC keyword. Output to this file is controlled by the SINGLEKEY keyword. The file contains the new keywords in text format. A typical use of this keyword is to convert the

input STOCHASTIC keywords, each defined for a statistical distribution, into new keywords defining a constant set to the median of the input distribution.

- **Values File.** The generated stochastic values can optionally be written to an output file. Output to this file is controlled by the DEBUG keyword. The file contains the generated values in text format.

### 38.3 Keyword Definitions for the STOCHASTIC Code

In general, the keywords can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

#### 38.3.1 DEBUG Keyword for STOCHASTIC

The DEBUG keyword is used to activate dumping of intermediate calculations to the report file, to write generated values to a file, and to write plot data to a file. It should be used sparingly because large output files can result. The following is this keyword's syntax:

```
DEBUG {DEFINITION}{STATISTICS}{STOCHASTIC="Quote 1" {COLUMN}}
      {PLOT="Quote 2" }
```

Multiple DEBUG keywords can be entered with combinations of modifiers, or a single card can be entered containing all of the modifiers. The modifiers can be entered in any order. Table 38.2 describes the modifiers associated with the DEBUG keyword.

**Table 38.2** Modifiers Associated with the DEBUG Keyword in STOCHASTIC

Modifier	Description
DEFINITION	The presence of the DEFINITION modifier on the DEBUG keyword will cause the definition of all stochastic variables to be written to the report file.
STATISTICS	The presence of the STATISTICS modifier on the DEBUG keyword will cause summary statistics for all generated variables to be computed and written to the report file.
STOCHASTIC	The presence of the STOCHASTIC modifier on the DEBUG keyword will cause all generated values to be written to the file identified in the associated quote string. If the modifier COLUMN is also present, the values will be written in a single column, otherwise, the values are written in comma-separated format with all values for a single variable on one line.
PLOT	The presence of the PLOT modifier on the DEBUG keyword will cause the empirical cumulative distribution function to be written to the file identified in the associated quote string.

The following sequence of keyword records illustrate the use of the DEBUG keyword:

```
DEBUG STOCHAST "Test_Median.Csv" COLUMN
DEBUG PLOT "Test_Plot_Median.csv"
DEBUG DEFINITI
DEBUG STATISTI
```

### 38.3.2 END Keyword for STOCHASTIC

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 38.3.3 REALIZAT Keyword for STOCHASTIC

The REALIZAT (or REALIZATION) keyword defines the number of realizations to generate. The following is this keyword's syntax:

```
REALIZATION value1
```

The integer value1 has a minimum value of 1. No specific maximum number of realizations is enforced, but large values may result in large output files. Values up to 5,000,000 have been verified using a small number of variables.

The following keyword record sets the number of realizations to 1000:

```
REALIZAT 1000
```

### 38.3.4 REPORT Keyword for STOCHASTIC

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [ "quote" ]
```

The name of the report file is entered in a quote string, which must be enclosed in double quotation marks. File names up to 200 characters long are supported, and optional path names can be included. The following is an example REPORT keyword record:

```
REPORT "G:\SAC\Sys\EC\Test.rpt"
```

### 38.3.5 SEED Keyword for STOCHASTIC

The SEED keyword sets the value for the seed for the random number generator. The following is this keyword's syntax:

```
SEED Value1
```

The value for Value1 must be an integer or real number in the range 1 to 999999. The following is an example keyword record:

```
SEED 344443
```

### 38.3.6 SINGLEKEY Keyword for STOCHASTIC

The SINGLEKEY keyword is used to output an optional file of data for a user-defined sample percentile for all stochastic variables. This option supports generation of median-value data sets for the inventory, vadose zone, and impact codes. The following is this keyword's syntax:

```
SINGLEKEY [PERCENT=Value1|MEAN] [FILE="quote1"]
```

Either the PERCENT modifier or the MEAN modifier is required. The numerical value associated with the modifier PERCENT is used to select the sample percentile for output. A value in the range 0 to 1 is allowed. A value of 0.5 will select the median generated value. The arithmetic mean of the generated values will be used for output if the MEAN modifier is used.

The quote string associated with the FILE modifier identifies the output file where the data will be written. The quote string must be enclosed in double quotation marks, and file names up to 200 characters long are supported. The following example indicates that keywords using median values for two stochastic variables are to be written to the file named "Tmp\_Key.Key":

```
SINGLEKEY PERCENT=0.5 File="Tmp_Key.key"  
STOCHASTIC KDSOIL "KdI129" 6 0.0 4.0 15.0 "Soil-water Kd for I129"  
STOCHASTIC KDSOIL "KdCs137" 9 6.908 0.692 TRUNCATE 0.01 0.99 "Kd Cs137"
```

This example results in the following two output keywords:

```
STOCHASTIC "KdI129" 1 5.9170E+00 "Soil-water Kd for I129"  
STOCHASTIC "KdCs137" 1 1.0002E+03 "Kd Cs137"
```

The following example indicates that keywords using the mean values for two stochastic variables are to be written to the file named "Tmp\_Key.Key":

```
SINGLEKEY MEAN File="Tmp_Key.key"  
STOCHASTIC KDSOIL "KdI129" 6 0.0 4.0 15.0 "Soil-water Kd for I129"  
STOCHASTIC KDSOIL "KdCs137" 9 6.908 0.692 TRUNCATE 0.01 0.99 "Kd Cs137"
```

This example results in the following two output keywords:

```
STOCHASTIC "KdI129" 1 6.3333E+00 "Soil-water Kd for I129"  
STOCHASTIC "KdCs137" 1 1.2289E+03 "Kd Cs137"
```

### 38.3.7 STOCHASTIC Keyword for STOCHASTIC

The STOCHASTIC keyword is used to enter the definition of a statistical distribution for stochastic variables. The following is this keyword's syntax:

```
STOCHASTIC ["Quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2} {"Quote2"}
```

- The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive, and embedded spaces are significant.

- The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive, and embedded spaces are significant.
- The optional entry for Quote2 is a description for the stochastic variable; the description can be up to 64 characters long.
- The entry for Dist\_Index must be an integer in the range 1 to 13 that identifies the index of a statistical distribution. The available statistical distributions are defined in Table 38.3.
- The word Parameters in the general syntax statement indicates the numerical values of parameters required for defining the statistical distribution.
- The additional modifier TRUNCATE can be used for all distribution types except 1, 3, and 10 (constant, discrete uniform, and user-defined). If TRUNCATE is entered, it must be followed by two values in the interval 0 to 1, inclusive. The lower value must be less than the upper value. These two values specify the tail probabilities at which to impose range truncation for the distribution. Truncation data must be entered after all of the other parameters that define the distribution. Further information on generation of stochastic variables is provided in Section 45.0.

**Table 38.3** Statistical Distributions Available in STOCHASTIC

Index	Distribution	Truncate	Parameters Required
1	Constant	No	Single value.
2	Uniform	Yes	Lower limit, upper limit.
3	Discrete Uniform	No	Smallest integer, largest integer.
4	Loguniform (base 10)	Yes	Lower limit, upper limit.
5	Loguniform (base e)	Yes	Lower limit, upper limit.
6	Triangular	Yes	Lower limit, mode, upper limit.
7	Normal	Yes	Mean, standard deviation.
8	Lognormal (base 10)	Yes	Mean, standard deviation of logarithms.
9	Lognormal (base e)	Yes	Mean, standard deviation of logarithms.
10	User Defined	Yes	Number of pairs, data for pairs of values (Prob( $X_i$ ), $X_i$ ).
11	Beta	Yes	Alpha, beta, lower limit, upper limit. The mean of the distribution would be $\alpha/(\alpha+\beta)$ if the limits were 0 and 1.
12	Log ratio	Yes	Mean, standard deviation (of derived normal), lower limit, upper limit.
13	Hyperbolic arcsine	Yes	Mean, standard deviation (of derived normal).

The following is an example STOCHASTIC keyword for a variable assigned a constant of 234.432:

```
STOCHASTIC "Unique1" 1 234.432 "Define a constant distribution"
```

The following is an example STOCHASTIC keyword for a bioconcentration factor that is normally distributed with a mean of 125 and a standard deviation of 5 for a frog exposed to  $^{14}\text{C}$ :

```
STOCHASTIC "BCFC14Frog" 7 125.0 5.0 "Example Distribution"
```

### 38.3.8 TITLE Keyword for STOCHASTIC

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the STOCHASTIC code."
```

### 38.3.9 USER Keyword for STOCHASTIC

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [ "quote" ]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```





## **39.0 Sto\_Med – Median Values for Stomp (Vadose Zone)**

### **39.1 Overview**

A run of the vadose zone flow and transport code STOMP with stochastic parameters set to the median value (50th percentile) of their range requires an input keyword file for the ESP code designed for a single realization. The input keyword file developed for the ESP code typically contains full stochastic distributions for the input variables. A sequence of computer codes is available that reads a full stochastic keyword file and writes the associated median-values keyword file.

#### **39.1.1 Location in the Processing Sequence**

Conversion of the stochastic keyword file into a median-values keyword file must occur before the vadose zone setup pass is performed by the ESP code.

#### **39.1.2 How the Code Is Invoked**

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. For purposes of discussion, let the stochastic keyword file be named “stochastic.key”, and let the median-values keyword file be named “median.key”. The sequence of three codes is invoked as follows:

```
inv_step1.exe "stochastic.key"  
stochastic.exe "Step1.Key"  
inv_step2.exe "stochastic.key" "median.key"
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

#### **39.1.3 Memory Requirements**

STO\_STEP1 and STO\_STEP2 have minimal memory requirements.

## **39.2 Computational Sequence**

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. Invocation of the three codes are described in the following paragraphs.

### **39.2.1 STO\_STEP1 Execution**

The first step in the processing sequence is to run the STO\_STEP1 code. This code reads the stochastic keyword file and writes a separate keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 2000 or XP), STO\_STEP1 executes in a DOS box. A run of STO\_STEP1 is initiated by entering the following command line:

```
STO_STEP1 "stochastic.key"
```

Under the Linux operating system STO\_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
sto_step1.exe "stochastic.key"
```

For these commands, “STO\_STEP1” or “sto\_step1.exe” is the name of the executable program, and “stochastic.key” is the name of the stochastic keyword file.

### 39.2.2 STOCHASTIC Execution

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the STO\_STEP1 code and writes a file named “Step1.Val” for use in the STO\_STEP2 code.

Under the Windows operating system (Releases 2000 or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC "Step1.Key"
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe "Step1.Key"
```

For these commands, “STOCHASTIC” or “stochastic.exe” is the name of the executable program, “Step1.Key” is the name of the input keyword file, and “Step1.Val” is the name of the output file of median values.

### 39.2.3 STO\_STEP2 Execution

The final step in the processing sequence is to run the STO\_STEP2 code. This code reads the stochastic keyword file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values keyword file.

Under the Windows operating system (Releases 2000 or XP), STO\_STEP2 executes in a DOS box. A run of STO\_STEP2 is initiated by entering the following command line:

```
sto_step2 "stochastic.key" "median.key"
```

Under the Linux operating system STO\_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
sto_step2.exe "stochastic.key" "median.key"
```

For these commands, “STO\_STEP2” or “sto\_step2.exe” is the name of the executable program, “stochastic.key” is the name of the stochastic keyword file, and “median.key” is the name of the median-values keyword file.

## **40.0 VOLEXT – Waste Stream Volume Summaries**

### **40.1 Overview**

The VOLEXT program extracts a sum of volumes by waste form for selected waste sites and selected years.

#### **40.1.1 Location in the Processing Sequence**

VOLEXT retrieves data from \*.res files built by the INVENTORY code. VOLEXT can be run anytime after the INVENTORY has been run.

#### **40.1.2 How the Code Is Invoked**

VOLEXT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 2000 or XP), VOLEXT executes in a DOS box. A run of VOLEXT is initiated by entering the following command line:

```
VOLEXT "Keyfilename"
```

Under the Linux operating system VOLEXT is executed through any of the following Bourne Shell or C Shell commands:

```
volect-1.exe "Keyfilename"
```

For these commands, “VOLEXT.EXE” or “volect-1.exe” is the name of the executable program, and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If VOLEXT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If VOLEXT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

#### **40.1.3 Memory Requirements**

VOLEXT has minimal memory requirements. A run using 700 waste sites, 40 waste forms, 15 analytes, and data spanning 70 years took less than 2 MB of memory.

### **40.2 File Definitions**

The VOLEXT program reads two input files and writes two output files. These files are described in the following sections.

### 40.2.1 Input Files

The VOLEXT program reads a control keyword file and a results file written by the INVENTORY code. The VOLEXT keyword file contains control information. An example file is provided in Table 40.1. Detailed definitions of the keywords are provided in Section 40.3.

**Table 40.1** Example Keyword File for VOLEXT

```
FILE REPORT "VolExt_Onsite.rpt"
TITLE "Extract volume sums for all sites from a *.RES file"
USER "Paul W. Eslinger"
VERBOSE
DEBUG
! File definitions
FILE RES      "inv1.res"          ! Input: The *.res file produced by INVENTORY
FILE OUTPUT "VolExt_Onsite.csv" ! Output : The matrix of summed volume values
! Years to include in the sum
YEARS ALL
! Sites to include in the sum
SITES LIST
  "4843"
  "100-B-15"
  "116-D-1B"
  "200 ETF"
  "216-A-36A"
  "218-E-14"
  "221-U"
  "241-SY-101"
  "UPR-100-N-1"
  "US_Ecology"
! End of the keywords
END
```

### 40.2.2 Output Files

The VOLEXT program writes two or more output files. One file is a report file that contains text information about the run of the code. It is not described further here; however, it should always be examined after a run of the code because error messages are written to the report file.

The other output file contains a matrix of comma-separated volume data. After a header line identifying the different waste forms, each line in the output volume file gives the volume of each waste form at a single site. The waste volumes at the site are summed over a user-specified year range. If the waste form does not exist at the site, a volume of zero is output. Output volume data for the two sites 118-D-6 and 118-KW-1 are shown in Table 40.2 for an inventory set containing 38 waste forms. The output lines in the file are too long to show in the table without wrapping; thus physical output lines are separated by a blank line in the table.

**Table 40.2** Excerpts from a VOLEXT–Generated Results File

```
Site,liquid,soiltu,soil,core,river,gas,cement,soilsn,soilln,rxcomp,soillf,
soil3n,cmntg3,cmntsf,cmntsn,cmntpe,glass,soil3f,cmntcu,cmntcs,store,
cmntfa,soileq,cmntuf,TRUslg,TRUtn,TRUttf,cmntct,TRUtsn,TRUtsf,HLW,off3,SF,
off4,off1,off2,off5

118-D-6, 0.000000E+00, 0.000000E+00, 1.000000E-03, 2.000000E+03,
0.000000E+00, 3.347280E+10, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00

118-KW-1, 0.000000E+00, 0.000000E+00, 1.000000E-03, 2.800000E+03,
0.000000E+00, 3.010631E+10, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00,
0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00, 0.000000E+00
```

### 40.3 Keyword Definitions for the VOLEXT Code

In general, the keywords can be entered in any order. The following restrictions apply to keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

#### 40.3.1 DEBUG Keyword for VOLEXT

The DEBUG keyword is used to activate dumping of intermediate calculations to the report file. It should be used sparingly because large output files can result. The following is this keyword's syntax:

```
DEBUG
```

#### 40.3.2 END Keyword for VOLEXT

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

### 40.3.3 FILE Keyword for VOLEXT

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. Two files must be defined for every run of the code using the FILE keyword. The quote string associated with the RES modifier provides the name of the inventory results file. This file must exist. The quote string associated with the OUTPUT modifier provides the file name for the output matrix of volume data. If an output file for volume data already exists, it will be overwritten when the code executes.

### 40.3.4 SITES Keyword for VOLEXT

The SITES keyword identifies the sites at which the calculations are to be performed for a single case. The following is this keyword's syntax:

```
SITES [ALL | LIST ["S1"] {"S2"} ... {"Sn"}]
```

All sites in the inventory results file are used when the modifier ALL is present. A list of specific sites can be defined by entering the modifier LIST followed by a list of site IDs, each enclosed in double quotation marks. The following example keyword extracts volume data for every site in the inventory results file.

```
SITES ALL
```

The following example keyword directs the code to extract volume data for a specific list of 10 different sites.

```
SITES LIST    "4843"      "100-B-15"    "100-B-3"    "100-B-5"    "100-B-8"  
              "100-C-3"    "100-C-6"    "100-D-23"    "100-D-24"    "100-D-29"
```

### 40.3.5 TITLE Keyword for VOLEXT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to the report file. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [ "quote" ]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the VOLEXT code."
```

### 40.3.6 USER Keyword for VOLEXT

The USER keyword is used to identify the user of the program. The user name will be written to the report file. The program will error terminate if the user name is not supplied. The following is this keywords syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public"
```

### 40.3.7 VERBOSE Keyword for VOLEXT

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. The following is this keyword's syntax:

```
VERBOSE
```

### 40.3.8 YEARS Keyword for VOLEXT

The YEARS keyword identifies the year range for which volume data are to be extracted and summed. The following is this keyword's syntax:

```
YEARS [ALL | [Year1] [Year2] ]
```

All data in the inventory results file are used in the volume sums when the modifier ALL is present. A range of years can be defined by entering two years without the ALL modifier. The following example keyword extracts volume data for every year in the inventory results file.

```
YEARS ALL
```

The following example keyword directs the code to extract volume data for the years 1950 through 1980 (inclusive):

```
YEARS 1950 1980
```





## **41.0 VZGRAB – Vadose Zone Data Grabber**

### **41.1 Overview**

The VZGRAB utility code extracts data associated with the vadose zone flow and transport model. Options include extracting control data from the ESD and VZGRAB keyword files, extracting data from vadose zone flow and transport output files, and formatting and exporting the extracted data for use in a spreadsheet or plotting program.

#### **41.1.1 How the Code Is Invoked**

VZGRAB only runs under the Linux operating system. VZGRAB is executed through the following Bourne Shell or C Shell command:

```
vzgrab-1.exe "ESDKeyfilename" "VZGRABKeyFileName"
```

For this command, “vzgrab-1.exe” is the name of the executable program, “ESDKeyfilename” is the name of the ESD keyword file providing overall control to the SAC simulation, and “VZGRABKeyFileName” is the name of the VZGRAB control keyword file. If VZGRAB is invoked without two filenames, the code will terminate execution after writing an error message to the standard output device. VZGRAB must be invoked from the root directory of a SAC analysis set, and the ESD keyword file name must be the local file name (no path information). The keyword file name can include path information.

#### **41.1.2 Memory Requirements**

VZGRAB uses dynamic memory allocation. Code runs using large data sets (1000 waste sites, 11 analytes, 100 realizations) may cause memory demands to exceed 2.1 gigabytes, thereby causing the run to error terminate. In this case, multiple runs using a smaller set of extractions are required to extract all of the data.

### **41.2 File Definitions**

The VZGRAB code reads two or more input files and writes one or more output files.

#### **41.2.1 Input Files**

The VZGRAB code reads the ESD keyword file, a VZGRAB-specific keyword file, and optionally many other files in the problem directory structure. Only the VZGRAB keyword file is discussed in this section. The example VZGRAB keyword file provided in Table 41.1 will extract release data for a single site, a single analyte, and five realizations.

**Table 41.1** Example Keywords for a Single Release Site for the VZGRAB Program

PATH "\file name\"	! Tell VZGRAB where to put the output
EXTRACT RELEASE	! Extract release to groundwater data
TIME ANNUAL TOTAL	! Sum to calendar years and entire simulation period
SITE "216-B-11A#B"	! Extract data for this site only
ANALYTE ID="H3"	! Extract data for this analyte only
REALIZATION 1 2 3 4 5	! Extract data for these five realizations
END	

The example VZGRAB keyword file provided in Table 41.2 will extract release data summed over all sites, a single analyte, and four realizations.

**Table 41.2** Example Keywords for Summing Over All Release Sites for the VZGRAB Program

PATH "\file name\"	! Tell VZGRAB where to put the output.
EXTRACT RELEASE	! Extract release to groundwater data.
TIME ANNUAL TOTAL	! Sum to calendar years & entire simulation period
SITE ALL SUM	! Extract data for all sites and sum over all sites
ANALYTE "Tc99"	! Extract data for this analyte only.
REALIZATION 1 2 3 4	! Extract data for these four realizations.
END	

The example VZGRAB keyword file provided in Table 41.3 will extract release data summed over a list of user-specified sites, a single analyte, and three realizations.

**Table 41.3** Example Keywords for Summing Over Selected Release Sites for the VZGRAB Program

PATH "\file name\"	! Tell VZGRAB where to put the output
EXTRACT RELEASE	! Extract release to groundwater data
TIME ANNUAL TOTAL	! Sum to calendar years & entire simulation period
LIST "216-B-3-2"	! Extract data only for the sites in this list.
LIST "216-B-3-3"	
LIST "216-B-3A"	
LIST "216-B-3B"	
LIST "216-B-3C"	
ANALYTE ID="U238"	! Extract data for this analyte only.
REALIZATION 1 2 3	!- Extract data for these three realizations only.
END	

## 41.2.2 Output Files

VZGRAB writes one or more output files. Each run creates a text log file that contains brief information about the job. For some combinations of inputs, the log file is the only output file written.

If annual release values are requested, a separate output file will be created for every analyte requested. These files reside in the directory defined using the PATH keyword and have names with the following structure: vz\_release\_gw\_ann\_XXXXXX.csv. The string XXXXXX is replaced by the analyte ID for each analyte. If cumulative releases are requested, a separate output file will be created for every analyte requested. These files reside in the directory defined using the PATH keyword and have names with the following structure: vz\_release\_gw\_cum\_XXXXXX.csv. The string XXXXXX is replaced by the

analyte ID for each analyte. The annual and cumulative release files are written in comma-separated variable format and are intended to be imported into a spreadsheet or plotting package for further analysis.

### 41.3 Keyword Descriptions for VZGRAB

All user instructions regarding data extraction are supplied via the VZGRAB keyword control file and communicated by means of keywords. Keywords may be provided in any order. However, all information after the END keyword will be ignored.

#### 41.3.1 ANALYTE Keyword for VZGRAB

The ANALYTE keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ ["quote 1"] {"qoute2"} ... {"quoteN"} | ALL]
```

Several ANALYTE keywords may be entered, or a single ANALYTE keyword with more than one ID can be used. The effect of multiple analyte keywords is cumulative. No quote strings are required if the modifier ALL is used because all analytes defined in the ESD keyword file for the analysis will be used. One or more quote strings containing the ID of analytes defined in the ESD keyword file must be entered if the ALL modifier is not used. Two examples of this keyword are the following:

```
ANALYTE "C14" "U238"  
ANALYTE ALL
```

#### 41.3.2 CUMULATIVE Keyword for VZGRAB

The CUMULATIVE keyword is used without any modifiers to direct VZGRAB to output extracted data in cumulative form. The default (if CUMULATIVE is absent from the VZGRAB control file) is to output extracted data as annual values. The CUMULATIVE keyword has no effect on time summations (such as the TIME TOTAL or TIME FROM year TO year commands). The following is this keyword's syntax:

```
CUMULATIVE
```

#### 41.3.3 END Keyword for VZGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

#### 41.3.4 EXTRACT Keyword for VZGRAB

The EXTRACT keyword defines what type of data is to be extracted. Several EXTRACT keywords with different modifiers may be included in one keyword file, but a single EXTRACT keyword with more than one modifier is also acceptable. The following is this keyword's syntax:

EXTRACT [INQUIRE | INFLUX | BALANCE | RELEASE | REMEDIATE | PARAMETERS |  
PROFILE | LIMIT | RESTART]

The modifiers for the EXTRACT keyword are defined in Table 41.4.

**Table 41.4** Modifiers Associated with the EXTRACT Keyword in the VZGRAB Control File

Modifier	Description
INQUIRE	Returns information about the SAC run, such as which analytes are defined and how many realizations were run
INFLUX	Amount of analyte (kg) or amount of radiation (Ci) for radioactive analytes influx and amount of fluid (m <sup>3</sup> ) influx to the Vadose Zone
BALANCE	Amount of analyte (kg) or amount of radiation (Ci) for radioactive analytes balance at SAC-wide times (use EXTRACT INQUIRE to determine balance times)
RELEASE	Amount of analyte (kg) or amount of radiation (Ci) for radioactive analytes released to the groundwater zone
REMEDiate	Amount of analyte (or amount of radiation) remediated from the Vadose Zone
PARAMETERS	Values of stochastic input parameters
PROFILE	Nodal STOMP concentrations reported by depth
LIMIT	Checks all STOMP results to detect time-step-limited terminations
RESTART	Checks the STOMP restart header files for correctness

Example uses of this keyword are the following:

```
EXTRACT RELEASE
EXTRACT BALANCE
EXTRACT RELEASE BALANCE RESTART
```

### 41.3.5 LIST keyword for VZGRAB

The LIST keyword defines a list of user-selected sites to group for summation. The following is this keyword's syntax:

```
LIST "quote1" ... "quoteN"
```

The SUM keyword should not be used if the LIST keyword is used. Several LIST keywords may be entered, or a single LIST keyword with more than one ID can be used; however, data from several LIST keywords define a single composite list of sites. Each quote string must contain a valid site ID from the ESD keyword file. The following example identifies five sites to sum as a group:

```
LIST "100-B-15" "100-B-3"
LIST "100-B-5" "100-B-8" "100-C-3"
```

### 41.3.6 NOHEADER Keyword for VZGRAB

The NOHEADER keyword is used without any modifiers to direct VZGRAB to not print header information to comma-separated variable (CSV) output files. The default is for VZGRAB to output headers to these files unless the NOHEADER command is present in the VZGRAB control file. This

option is useful for automated graphics production with third-party graphics software (such as IGOR Pro™). The following is this keyword's syntax:

```
NOHEADER
```

### 41.3.7 PATH keyword for VZGRAB

The PATH keyword defines the path to the directory where the extracted results are stored. The following is this keyword's syntax:

```
PATH "path name"
```

The path name must be entered in a quote string and should end with the delimiter "/". An example use of this keyword is the following:

```
PATH "/home/ANALYSIS4/CA1_median/vzgrab/"
```

### 41.3.8 OUTPUT keyword for VZGRAB

The OUTPUT keyword for VZGRAB is obsolete. If present, it will be ignored.

### 41.3.9 REALIZAT keyword for VZGRAB

The REALIZAT (or REALIZATION) keyword defines which realization or realizations to report. The following is this keyword's syntax:

```
REALIZATION [N1 {N2} ... {Nk} | ALL]
```

Several REALIZAT keywords may be entered, or a single REALIZAT keyword with more than one number can be entered. If the modifier ALL is used, all realizations defined in the ESD keyword file will be used. Otherwise, one or more realization numbers must be entered. Two example uses of this keyword are the following:

```
REALIZAT ALL ! Use all realizations  
REALIZAT 5 2 7 100 ! Define 4 realizations explicitly
```

### 41.3.10 SITE Keyword for VZGRAB

The SITE keyword is used to define one or more sites to report on. The following is this keyword's syntax:

```
SITE ["quote1" {"quote2"} ... {"quoteN"} | ALL SUM]
```

The LIST keyword should not be used if the SITE keyword is used. Data for each designated site will be reported separately if the modifier SUM is not used. If the modifiers ALL and SUM are used, then the data will be reported as the sum over all sites in the ESD keyword file. The LIST keyword must be used to sum data for a subset of sites. The first example below reports data individually for five waste sites. The second example below reports data as the sum over all sites:

```
SITE "100-B-15" "100-B-3" "100-B-5" "100-B-8" "100-C-3"  
SITE ALL SUM
```

#### 41.3.11 TIME keyword for VZGRAB

The TIME keyword defines how to present the data with respect to time. The following is this keyword's syntax:

```
TIME [{ANNUAL} {TOTAL} {FROM year1 year2}]
```

Data will be summed on a calendar year basis if the modifier ANNUAL is present. Data will be summed for the entire simulation period if the modifier TOTAL is present. Data will also be summed over a range of years from year1 to year2 if the FROM modifier is present. Three example uses of this keyword are the following:

```
TIME ANNUAL  
TIME ANNUAL TOTAL  
TIME ANNUAL FROM 1944 TO 3000
```

Although both TOTAL and FROM modifiers can be used in a single run of the code, the data for both options are written to the same file in an interlaced fashion. Using both the TOTAL and FROM options in the same run of the code is not recommended.

## 42.0 VZSWAP – Vadose Zone Data Replacement

### 42.1 Overview

The VZSWAP utility code provides a convenient means to swap third-party vadose zone release results produced outside the SAC framework (for example, from site-specific performance assessment analyses) into or out of a SAC assessment.

#### 42.1.1 Location in the Processing Sequence

The VZSWAP code can be executed after the SAC vadose zone model has been run.

#### 42.1.2 How the Code Is Invoked

Before invoking the VZSWAP code, the user must place the third-party release to be swapped for SAC results in the appropriate /vadose subdirectories in files named “release.3RD” and prepare a swapfile that lists the /vadose subdirectories on which to operate.

The VZSWAP code is then invoked from this command and two arguments:

```
vzswap-1.exe swapfile [3RD|SAC|SUM]
```

The executable name is “vzswap-1.exe”. The *swapfile* argument is the name of a text file containing a list of vadose zone directories to perform the swap operation. The second argument may be either 3RD, SAC, or SUM. The modifier 3RD is used to direct VZSWAP to place third-party release results into the “release” file. The SAC modifier directs VZSWAP to restore original “release” file contents generated by SAC. The SUM modifier directs VZSWAP to sum both the third-party release and the SAC release into a new summed release record.

To help prompt the user, help is provided if the second command argument is “help”. For example,

```
vzswap-1.exe help
```

#### 42.1.3 Memory Requirements

VZSWAP requires a trivial amount of memory.

### 42.2 File Definitions

The following files are utilized by VZSWAP:

- **swapfile** – an ASCII text file produced by the user that specifies the directories under the /vadose directory of a SAC assessment that will be operated on by VZSWAP
- **release** – the original release file produced by SAC using the STOMP code in a given /vadose subdirectory
- **release.SAC** – the file in which VZSWAP will create a copy of the original “release” file found in a given /vadose subdirectory and store it as ‘release.SAC’

- **release.3RD** – the file in which the 3rd-party release results are provided, in the same format as a STOMP “release” file (except the #SAC header information is not required; VZSWAP will insert these lines).



## 43.0 WTRTBL and RDWTRTBL – Hydraulic Head Extractor for CFEST Results

### 43.1 Overview

WTRTBL is a utility code that reads CFEST binary files for a completed transient flow solution and writes a pair of files that ESP uses for fast access to the water table elevation at CFEST nodes nearest each vadose zone site when preparing STOMP input files.

For context, STOMP input files are written by ESP to include transient Dirichlet aqueous phase pressure at the lower boundary that serves to maintain the water table within the vadose zone domain at the elevations passed from the CFEST flow solution using WTRTBL-generated files. ESP also includes in the STOMP input files a list of nodes and times that indicate the position of the water table to the nearest STOMP node; all mass at or below the current “release plane” node are extracted and passed to VZDROP. This provides the means for dynamic linking of a transient water table with the vadose zone solution.

#### 43.1.1 Location in the Processing Sequence

WTRTBL is designed to be executed in standalone fashion once a CFEST flow template is completed. The files it produces (“watertable.bin” and “watertable.idx”) should be copied with other files that constitute the CFEST flow template for inclusion in a SAC production run. ESP will expect to find these two files in the template directory indicated by the /cfest/stochastic.key file.

#### 43.1.2 How the Code Is Invoked

WTRTBL and RDWTRTBL run only under the Linux operating system. WTRBL is invoked with no command line arguments; it must be invoked from the CFEST problem directory for the results desired:

```
wtrtbl-1.exe
```

RDWTRTBL is invoked with the following command-line calling arguments:

```
rdwtrtbl-1.exe nodeID
```

where nodeID is an integer indicating the ID of the desired CFEST node.

RDWTRTBL must be invoked in the same directory as the files “watertable.idx” and “watertable.bin” are located that are being probed.

### 43.2 File Structure

The WTRTBL code reads five input files and writes two output files.

#### 43.2.1 Input Files

The WTRTBL code reads five CFEST files as follows:

- **cfest.ctl** – The CFEST control file that contains data on input filenames, output control switches, and solver parameters, as well as several simulation switches.
- **cfest.b01** – This CFEST file stores node and element data, material properties, initial conditions, and fixed boundary condition data.
- **cfest.b07** – This CFEST file stores head and concentration data for each time step.
- **cfest.b08** – This CFEST file contains time step information and flux boundary condition data.
- **cfest.zzz** – This CFEST file contains mass balance data.

### 43.2.2 Output Files

The hydraulic heads for all time steps for the node with node identification number `nodeID` will be exported to a file named “head.*nnn*” where *nnn* is the value given by `nodeID`.

WTRTBL produces two files:

- “watertable.bin” is a direct-access binary file containing one record per CFEST node (all nodes, not just surface nodes). Each record contains the hydraulic head for each CFESTS time step.
- “watertable.idx” is an ASCII text file containing the necessary information to access the records in “watertable.bin”

These two files will provide ESP with all the information needed for fast access to water table histories by node ID when building STOMP “input” files.

The file “watertable.bin” is a direct-access binary file. It will contain the same number of records as there are nodes in the CFEST flow solution it is extracted from. Each record will contain a set of hydraulic heads, one per time step for the same number of time steps are in the CFEST flow solution it is extracted from, in increasing time order. The hydraulic heads are stored in double-precision; REAL(KIND=8).

The file “watertable.idx” is an ASCII text file. It contains one value per line. The contents are as follows:

**Table 43.1** Contents of the Output File “watertable.idx” for the WTRTBL Program

Line number	Content	Description
1	numTimes	Number of CFEST time steps
2	numNodes	Number of CFEST nodes
3...(3+numTimes)	time(1:numTimes)	Time at end of each CFEST time step (days since January 1, 1944)
(3+numTimes+1)... (3+numTimes+numNodes)	nodeID(1:numNodes)	Node ID for each node in CFEST order

Both RDWTRTBL and ESP’s MODSTP routine are able to use the information in “watertable.idx” to support reading the contents of “watertable.bin”.

Because WTRTBL writes the hydraulic heads to a binary file, a related utility code RDWTRTBL is also provided that allows the user to probe “watertable.bin” to extract all CFEST hydraulic heads for a single node to an ASCII text file for checking purposes.



## 44.0 Keyword Language Syntax

Each line of a keyword data file is parsed into numeric and character data. These are interpreted to set up control information and define input parameters. An input line can contain up to 2048 characters of information. Quote strings are limited to 200 characters in length.

Every line of the input data file is considered a keyword record, continuation record, or a comment record. Keyword records contain a keyword beginning in column 1. The keyword is used to determine the purpose of the subsequent data. Continuation records are used when a keyword record requires too much data to be placed on one line. Comment lines are ignored by the reading software but are useful for annotating the input file.

The information from each keyword record and subsequent continuation lines is moved into storage arrays. Data that can be deciphered as numeric values are placed into a numeric array. Other data are classified either as “secondary keywords” (called “modifiers”) or “quote strings.” Secondary keywords are stored as character images in an array. All such keywords or modifiers read from the input file are changed to uppercase before being stored. Quote strings are text strings that are enclosed in double quotation marks. These are stored exactly as they are read from the input file.

### 44.1 Keyword Records

Keyword records start in column 1 with any letter from A to Z, in either upper- or lowercase. The first eight letters of a keyword are stored in a variable and are used by the modeling software to determine the actions desired by the program user. All subsequent lines of text that do not have an alphabetic character or comment character in column 1 are treated as continuation lines. The following is an example keyword record (where SAMPLEKEY starts in column 1):

```
SAMPLEKEY 2 0 500 1 100
```

The word SAMPLEKEY is the keyword. The numbers 2, 0, 500, 1, and 100 are numeric data.

### 44.2 Continuation Records

Continuation records start with any valid separator character (except a double quotation mark). These are treated as additional data to the previous keyword record. Section 44.4.2 identifies valid separator characters. The combined data on a keyword line and on the subsequent continuation line(s) are treated as a single block of information. All numeric values and character strings on those lines are used as input data relevant to the keyword of the keyword line. The two following keyword entries contain the same information:

```
SAMPLEKEY 2 0 500 1 100
```

```
SAMPLEKEY 2 0  
500 1 100
```

## 44.3 Comment Records

Any line with the characters \$, ! or / in column 1 will be treated as a comment record. These lines are ignored by the input data record reader. Both the \$ and the ! can also be used to signify in-line comments (not in column 1). Any information that follows a \$ or a ! will be ignored. The / character indicates a comment only if it is the first character on the input line. The following are some examples of comment lines:

```
$This is a comment line  
/This is a comment line  
!This is a comment line
```

The following are some examples of in-line comments:

```
SAMPLEKEY 3 4.0 5.0 !Trailing information is ignored after the !  
SAMPLEKEY 3 4.0 5.0 $Trailing information is ignored after the $
```

## 44.4 Input Data Handling

Each line of the input is read and parsed into numeric and character values. All numeric values are converted to real numbers (as opposed to FORTRAN integer). All data that cannot be interpreted as numeric information are stored as character values.

Numeric data can include a leading sign (+ or -), integer characters 0 through 9, a decimal point, and an exponent indication ("E" or "e"). The FORTRAN "double precision" exponent indicator "D" is not valid. A maximum of 10 digits is allowed when entering numeric data.

Secondary keywords, or modifiers, are character strings that could not be interpreted as numeric values. These are converted to uppercase, where necessary, and stored in an array. The number of secondary keywords that are moved into the array is stored for internal use.

Only the first eight characters of any keyword are significant. Keywords fewer than eight characters long are left justified and blank-filled.

### 44.4.1 Quote Strings

Quote strings are strings of literal text that must be used exactly as given in the input line. They are enclosed by double quotation marks and are typically used for passing file names into a program. Only the first 200 characters of a quote string are saved. Each quote string must begin and end on a single line of the input file. When an unclosed quote is encountered, an implied quote is created at the end of that line. The following is an example of quote string usage:

```
FILE "c:\apps\human\test.dat"
```

#### 44.4.2 Data Separators

Keywords, numeric data, and secondary keywords must be separated by any one of the following data “separators”: space character, comma, equal sign, colon, semicolon, left parenthesis, right parenthesis, single quotation mark, double quotation mark, and tab character.

Also, any character with a ASCII character storage code of less than 10 is treated as a separator character. This is used mainly to identify the ASCII tab character as a data separator. Double quotation marks are used differently than the other separators. They indicate text strings that are stored without conversion by the program. As an illustration of the use of separator characters, the following keyword records all contain and convey the same information:

```
SAMPLEKEY 3 4.5 5.6 6.7  
SAMPLEKEY 3 (4.5,5.6,6.7)  
SAMPLEKEY 3 ( 4.5=5.6'6.7 )  
SAMPLEKEY 3:4.5 5.6:6.7
```





## 45.0 Stochastic Variable Generation

Many of the codes in SAC, Rev. 1, generate values for stochastic variables. All of the codes use the same suite of statistical routines to do this generation. The following are some major considerations for this process:

- Each distribution is generated using the Probability Integral Transformation method (Mood et al. 1974, p. 202).
- The uniform number generator uses a linear congruential method (Lewis et al. 1969).
- Stratified sampling is used when the number of values to be generated is greater than 1.
- Most distributions may be truncated between two limits that are specified as limits in the uniform domain on the interval 0 to 1.
- The user may specify a cumulative distribution function in the form of a table of values.
- Information about a stochastic variable is linked to a unique character ID. Access to all information about the variable is available through use of the variable ID.

The following are the available statistical distributions:

- Constant value
- Uniform distribution between two limits
- Discrete uniform distribution on a set of contiguous integers
- Loguniform (base 10) distribution between two limits
- Loguniform (base e) distribution between two limits
- Triangular distribution defined using a lower limit, mode, and an upper limit
- Normal distribution with a mean and standard deviation
- Lognormal (base 10) distribution specified by the mean and standard deviation of the logarithms of the data
- Lognormal (base e) distribution specified by the mean and standard deviation of the logarithms of the data
- User-specified cumulative distribution function input as a table of probabilities and exceedance values
- Beta distribution that can be shifted and scaled from the standard (0,1) interval
- Log-ratio from a normal distribution
- Hyperbolic arcsine from a normal distribution.

### 45.1 Keywords Defining Stochastic Variables

Depending on the utility code, stochastic variables can be defined for STOCHAST, KDSOIL, DILUTE, POROSITY, SATURATI, HYDRAULI, SORPTION, and RETARDAT keywords. The following

general description is presented for STOCHAST(IC) keyword, although it applies to each of the other keywords. The following is this keyword's syntax:

```
STOCHASTIC [{ID=} "Quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2}
[{LABEL=} "Quote2"] {UNITS="quote3"}
```

- The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive, and embedded spaces are significant. It is sometimes useful to make the character string some combination of a variable name and other data so that it can be recreated easily when stochastic data is needed.
- The entry for Quote2 is an optional description for the stochastic variable that can be up to 64 characters long and is used for output labeling purposes.
- The entry for Dist\_Index must be an integer in the range 1 to 13 that identifies the index of a statistical distribution. The statistical distributions are defined in Table 45.1.
- The word Parameters in the general syntax statement indicates the numerical values of parameters required for defining the statistical distribution.
- The additional modifier TRUNCATE can be used for all distribution types except 1, 3, and 10. If TRUNCATE is entered, it must be followed by two values in the interval 0 to 1, inclusive. The lower value must be less than the upper value. These two values specify the tail probabilities at which to impose range truncation for the distribution. Truncation data must be entered after all of the other parameters that define the distribution.

**Table 45.1** Common Statistical Distributions Available in SAC Codes

Index	Distribution	Truncate	Parameters Required
1	Constant	No	Single value.
2	Uniform	Yes	Lower limit, upper limit.
3	Discrete Uniform	No	Smallest integer, largest integer.
4	Loguniform (base 10)	Yes	Lower limit, upper limit.
5	Loguniform (base e)	Yes	Lower limit, upper limit.
6	Triangular	Yes	Lower limit, mode, upper limit.
7	Normal	Yes	Mean, standard deviation.
8	Lognormal (base 10)	Yes	Mean, standard deviation of logarithms.
9	Lognormal (base e)	Yes	Mean, standard deviation of logarithms.
10	User Defined	Yes	Number of pairs, data for pairs of values (Prob( $X_i$ ), $X_i$ ).
11	Beta	Yes	Alpha, beta, lower limit, upper limit. The mean of the distribution would be $\alpha/(\alpha+\beta)$ if the limits were 0 and 1.
12	Log ratio	Yes	Mean, standard deviation (of normal), lower limit, upper limit.
13	Hyperbolic arcsine	Yes	Mean, standard deviation (of normal).

The following is an example STOCHASTIC keyword for a variable assigned a constant of 234.432:

```
STOCHASTIC "Unique1" 1 234.432 "Define a constant distribution"
```

The constant can take any value.

The following is an example STOCHASTIC keyword for a variable assigned a uniform distribution on -2 to 7:

```
STOCHASTIC ID="Unique2" 2 -2.0 7 "Uniform distribution on -2 to 7"
```

The two limits can take any values as long as the second value is strictly greater than the first value. The following is an example stochastic keyword for a variable assigned a discrete uniform distribution on the integers 6 to 70:

```
STOCHASTIC "Unique3" 3 6 70 "Discrete uniform distribution on 6 to 70"
```

The two limits must be integers where the second integer is strictly greater than the first integer.

The following is an example STOCHASTIC keyword for a variable assigned a loguniform (base 10) distribution on the interval  $10^{-7}$  to  $10^{-3}$ :

```
STOCHASTIC ID="Unique4" 4 1.0E-7 1.0E-3  
LABEL="Define a loguniform (base 10) variable on 0.0000001 to 0.001"
```

The two limits must both be greater than zero, and the second limit must be greater than the first limit.

The following is an example STOCHASTIC keyword for a variable assigned a loguniform (base e) distribution on the interval  $10^3$  to  $10^6$ :

```
STOCHASTIC "Unique5" 5 1.0E3 1E+6  
"Define a loguniform (base e) distribution on 1000 to 1000000"
```

The two limits must both be greater than zero, and the second limit must be greater than the first limit.

The following is an example STOCHASTIC keyword for a variable assigned a triangular distribution with a minimum of 2, a mode of 3, and a maximum of 7:

```
STOCHASTIC "Unique6" 6 2 3 7 "Triangular distribution on (2,3,7)"
```

The three values that define the triangular must all be different, and they must be entered in increasing order.

The following is an example STOCHASTIC keyword for a bioconcentration factor that is normally distributed with a mean of 125 and a standard deviation of 5 for a frog exposed to  $^{14}\text{C}$ :

```
STOCHASTIC "BCFC14Frog" 7 125.0 5.0 "Example normally distributed frog"
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword would define a different stochastic variable from the one just entered because the identification string (Quote1) is case sensitive:

```
STOCHASTIC "BCFC14FROG" 7 125.0 5.0 "Example normally distributed frog"
```

The following keyword entry would define a lognormal (base 10) distribution where the mean and standard deviation (of the logarithms) are  $-2.0$  and  $0.5$ :

```
STOCHASTIC "Unique8" 8 -2 0.5 "Lognormal (base 10) variable"
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword entry would define a lognormal (base e) distribution where the mean and standard deviation (of the logarithms) are  $-2.0$  and  $0.5$ . In addition, the lognormal distribution will be truncated between the lower  $0.025$  and upper  $0.99$  probabilities.

```
STOCHASTIC "Unique9" 9 -2 .5 TRUNCATE 0.025 0.99  
"Example for a truncated lognormal variable"
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword entry illustrates the use of the user-defined distribution (distribution type 10). This example entry uses seven pairs of values. The first pair of numbers uses a probability of 0 to define the lower limit of the distribution at  $8.4\text{E-}7$ . The last pair of numbers uses a probability of 1 to define the upper limit of the distribution at  $1.73\text{E-}6$ . The other values are associated with the probability levels of .025, .167, .5, .833, and .975. The probability data and distribution percentiles must be entered in strictly increasing order.

```
STOCHASTIC "Sr90Con" 10 7  
0 8.40E-7  
2.50E-02 9.20E-7  
1.67E-01 1.06E-6  
5.00E-01 1.21E-6  
8.33E-01 1.37E-6  
9.75E-01 1.58E-6  
1 1.73E-6
```

The first pair of numbers uses a probability of 0 to define the lower limit of the distribution. The last pair of numbers uses a probability of 1 to define the upper limit of the distribution. The intervening pairs define probability levels and the associated data values. The probabilities and data values must be entered in strictly increasing order.

The following keyword entry would define a beta distribution with parameters 1.1 and 2.1 on the interval (0,1):

```
STOCHASTIC "Unique11-1" 11 1.1 2.1 0.0 1.0  
"Beta (1.1,2.1) on the interval 0,1"
```

Let the first parameter be denoted by  $\alpha$  and the second parameter be denoted by  $\beta$ . The mean of the beta distribution would be  $\alpha / (\alpha + \beta)$  if the limits were 0 and 1. Both  $\alpha$  and  $\beta$  must be greater than zero. The lower limit must be less than the upper limit.

The following keyword entry would define a beta distribution with parameters 1.1 and 2.1 but on the interval  $-2$  to  $4$ :

```
STOCHASTIC "Unique11-2" 11 1.1 2.1 -2.0 4.0  
"Beta (1.1,2.1) on the interval (-2,4)"
```

The following keyword entry would define a log ratio distribution from a normal (-1.459,1.523) distribution on the interval -5.756 to 4.33:

```
STOCHASTIC "Test1203" 12 -1.459 1.523 -5.756 4.330  
"Log ratio from Normal(-1.4,1.5) on (-5.756,4.330)"
```

The entry for the normal standard deviation (a value of 1.523 in this example) must be greater than zero. The last two numerical values define the interval for the generated values, so the lower limit must be smaller than the upper limit.

The following keyword entry would define a hyperbolic arcsine distribution from a normal (0.189,0.146) distribution:

```
STOCHASTIC "Test1302" 13 0.189 0.146  
"Hyperbolic Arcsine from Normal(0.189,0.146)"
```

The entry for the normal standard deviation (a value of 0.189 in this example) must be greater than zero.

## 45.2 Probability Concepts

The distribution of a continuous random variable  $X$  (the term *continuous* indicates that the random variable is defined over a continuum of values) is completely described by its probability density function,  $f(x)$ . The interpretation of the probability density function is that the area under  $f(x)$ , for an interval  $a < x < b$ , equals the probability that the random variable,  $X$ , will fall in the interval  $(a,b)$ , denoted  $P[a < X < b]$ . One cannot make the statement  $P[X=t]$ , because there is no area under the probability density function at any given point. Two axioms of probability theory (Mood et al. 1974, p. 22) are that the probability of any event is between zero and one, and the integral of the probability density function over the entire support (the interval  $[L,U]$ ) of  $X$  equals 1. The integral of the probability density function from the lower bound  $L$  to some value  $x$  (suppose it is less than the upper bound  $U$ ) represents the probability that  $X$  will be observed in the interval  $(L,x)$ . This integral operation defines the cumulative distribution function for the random variable  $X$ . The cumulative distribution function is denoted by  $F(x)$  (the capital  $F$  for the cumulative distribution function corresponds to the lowercase  $f$  for the probability density function) and mathematically is represented by the following:

$$F(x) = \int_L^U f(s)ds$$

The inverse cumulative distribution function,  $[F^{-1}(\bullet)]$ , is single-valued if  $x$  is in the interval  $(L,U)$ . Hence if  $p' = F(x')$  is known, in theory  $x' = F^{-1}(p')$  can be obtained.

## 45.3 Probability Integral Transform Method

Generation of a random variable from a given distribution typically involves the use of information either about  $f$  or  $F$ . There are two philosophical approaches to generating random numbers: exact methods and approximate methods. The algorithms embedded in SAC employ exact methods. Exact methods can be

further categorized into probability integral transform methods and functional methods. The probability integral transform method is employed in SAC.

In the probability integral transform method, the random variable of interest is expressed as a function of a  $U(0,1)$  random variable, where  $U(0,1)$  denotes the continuous random variable ranging uniformly over the interval  $(0,1)$ . The probability density function of the uniform random variable is  $g(u)=1$  if  $u \in (0,1)$  and is zero elsewhere. The cumulative distribution function for this random variable takes the exceedingly simple form  $G(u)=u$ . It can be shown that any cumulative distribution function evaluated at a random value  $X$  (instead of being evaluated at a known value  $x$  as in the previous discussion) is distributed uniformly over the interval  $(0,1)$  (Mood et al. 1974, p. 202). Therefore, given a realization  $u$  of the  $U(0,1)$  random variable and a selected statistical distribution (known cumulative distribution function), one can set  $u=F(x)$  and solve to obtain  $x=F^{-1}(u)$ . The value  $x$  thus obtained is a random realization from the selected statistical distribution.

In principle, one can obtain an exact solution for  $x$  given any specific cumulative distribution function and value  $u$ . In reality, there exist some distributions, such as the normal and beta distributions, for which no closed-form analytical expression for  $F^{-1}$  exists, and hence approximation methods must be applied.

The probability integral transform method allows efficient sampling from a subregion of the interval  $(L,U)$ , such as  $(c,d)$ , where  $L < c < d < U$ . In this case one would find the corresponding interval in the uniform domain, say  $(c',d')$ , and sample uniformly over that interval, by sampling from the rescaled uniform distribution [for example,  $u'=(d-c)u+c$ ] and then obtaining  $x$  as usual using  $x=F^{-1}(u')$ . The rescaled uniform distribution takes the form  $g(u)=1/(d'-c')$  for  $u \in (c',d')$  and is zero elsewhere. For any distribution with probability density function  $f(x)$  and cumulative distribution function  $F(x)$ , the probability density function, under truncation to the interval  $(c,d)$ , is  $f_T(x) = f(x)/[F(d)-F(c)]$ . The divisor ensures that  $f_T(x)$  integrates to unity.

## 45.4 Stratified Sampling

Stratified sampling can easily be implemented when generating random deviates using the probability integral transform method. This is accomplished by dividing the uniform interval  $(0,1)$  into subintervals, or strata, and sampling a specified number of times within each stratum, each time obtaining the corresponding value of  $x$ . Within SAC, the strata intervals are assigned equal probability, and exactly one value is sampled within each stratum. The method generates samples from each stratum and then randomly shuffles the entire set of realizations using a variation of the Quicksort algorithm (Hoare 1961). The primary purpose of stratified sampling is to achieve more evenly spaced (in a probability sense) samples from the distribution of a random variable than would result from randomly sampling over the whole range of the distribution. Iman and Conover (1982) have shown that stratified sampling can result in more efficient estimation of simulation results for a variety of estimators than when using simple random sampling.

Two steps are used to obtain a stratified sample  $\{s_i, i = 1, 2, \dots, N\}$  of size  $N$  for the uniformly distributed variable  $S$ . First, generate values for  $s_i$  using the following equation:

$$s_i = \frac{(n-1) + u_i}{N}$$

where  $u_i$  is a uniformly distributed number between 0 and 1 generated using simple random sampling. These values satisfy the relationship  $0 < s_i < s_{i+1} < 1$ . In the second step, the  $s_i$  are reshuffled to a random order. This reshuffling can be achieved by generating a new sequence of uniformly distributed random numbers,  $a_i$ ,  $i = 1, 2, \dots, N$  using simple random sampling. The set of values  $\{s_i\}$  are then reordered so they have the same rank order as the corresponding  $a_i$ .

## 45.5 Generation Algorithms

Table 45.1 summarizes the statistical distributions available in the SAC codes. The following paragraphs describe the generation algorithms.

### 45.5.1 Algorithm for the Uniform Distribution

Algorithms that generate truly random uniform numbers do not exist, although many algorithms generate pseudo-random deviates (hereafter loosely referred to as random numbers). The selection of a random number generator is based on four considerations: 1) computer implementability, 2) degree of independence within a sequence of deviates, 3) periodicity or cyclic length of a sequence, and 4) uniform coverage of sequences (occurrence) over the interval (0,1), the square (0,1) X (0,1), etc., up to the hypercube (0,1) in  $k$  dimensions.

Commonly used random number generation techniques are linear congruential methods (Park and Miller 1988). The SAC codes use a linear congruential random number generator. The linear congruential generator generates random integers from an algorithm of the form  $S_i = (A \cdot S_{i-1} + C) \text{ mod } M$ , where  $S_i$  is the  $i$ th generated random integer,  $A$  and  $C$  are constants,  $M$  is the modulus of the generated integers, and  $\text{mod}$  denotes the remainder function. These integers are converted to approximate uniform (0,1) numbers by the division  $U_i = S_i / M$ .

The period of a sequence  $\{U_i\}$  of generated deviates is the minimal value  $k$  such that  $U_i = U_{i+k}$  (this occurs independent of  $i$  for linear congruential generators). It can be shown that the period of any congruential generator does not exceed  $M$ . Therefore, if one is generating many uniform deviates, it is desirable that  $M$  be large. The performance of each congruential generator (each choice of  $A$ ,  $C$ , and  $M$ ) can thus be examined with respect to criteria proceeding from the four considerations given above.

**Generation Algorithm:** The SAC implementation for generating uniform random variables uses a linear congruential generator with  $A=16807$ ,  $C=0$ , and  $M=2147483647$ . These choices yield a sequence  $\{U_i\}$  that 1) is implementable on a 32-bit computer without machine language coding, 2) is sufficiently independent on an element-by-element basis, 3) possesses a long cycle (period), and 4) has a reasonable degree of coverage over all hypercubes of dimension less than  $k$ . These conclusions proceed from results from tests described in Fishman and Moore (1986). The generation algorithm uses two steps;

$$S_i = A \times S_{i-1} \text{ Mod}(M)$$

$$U_i = S_i / M$$

Any value,  $x$ , generated from the uniform (a,b) distribution in the SAC codes makes use of a value,  $y$ , from the  $U(0,1)$  distribution. The value  $y$  is first generated, and then  $x$  is evaluated as  $x=a+(b-a)y$ . The uniform (a,b) distribution will be denoted by  $U(a,b)$ .

Cumulative Distribution Function: The CDF for a U(a,b) random variable takes the form:

$$F(x) = \begin{cases} 0 & x < a \\ \left[ \frac{x-a}{b-a} \right] & a \leq x \leq b \\ 1 & x > b \end{cases}$$

Expected Value: The expected value of a U(a,b) random variable is (a+b)/2

Variance: The variance of a U(a,b) random variable is (b-a)<sup>2</sup>/12

Median: The median of a U(a,b) random variable is (a+b)/2.

### 45.5.2 Algorithm for the Discrete Uniform Distribution

Generation Algorithm: The probability density function for the discrete uniform distribution is f(x)=1/N for each of the N integers ranging in the interval L to U. The generation algorithm for the discrete uniform distribution is

$$F^{-1}(u) = L + \text{int}[u(U - L + 1)]$$

where the int(·) function returns the integer portion of its argument.

Cumulative Distribution Function: The cumulative distribution function of a discrete uniform random variable on the interval (a,b) is

$$F(x) = \begin{cases} 0 & x < a \\ \frac{1}{(b-a+1)} \sum_{i=a}^b I(a \leq i \leq x) & a \leq x \leq b \\ 1 & x > b \end{cases}$$

Expected Value: The expected value of a discrete uniform random variable on the interval (a,b) is

$$E(X) = \frac{b(b+1) - a(a+1)}{2(b-a+1)}$$

Variance: The variance of a discrete uniform random variable on the interval (a,b) is

$$V(X) = \frac{x(x+1)(2x+1) - (a-1)a(2a+1)}{6(b-a+1)} - \left[ \frac{b(b+1) - a(a+1)}{2(b-a+1)} \right]^2$$

Median: The median of a discrete uniform random variable on the interval (a,b) is the integer closest to (a+b)/2.

### 45.5.3 Algorithm for the Loguniform Distribution

Generation Algorithm: The probability density function for the loguniform random variable of base b is:



$$f(x) = \frac{I(b^c < x < b^d)}{x(d - c) \bullet \ln(b)}$$

for  $-\infty < c < d < \infty$ , where  $I$  is an indicator function (0 if false, 1 if true),  $b$  is the logarithm base (either 10 or the natural constant  $e$ ), and  $\ln(b)$  denotes the natural logarithm of  $b$ .

The inverse cumulative distribution function algorithm used to generate a value,  $x$ , from the loguniform distribution first generates a value,  $y$ , from the  $U(c,d)$  distribution and then evaluates the expression  $x=b^y$ .

Cumulative Distribution Function: The CDF algorithm for a loguniform  $(c,d)$  random variable is

$$F(x) = \begin{cases} 0 & x < b^c \\ \left[ \frac{\ln(x) - c \ln(b)}{(d - c) \ln(b)} \right] & x \in [b^c, b^d] \\ 1 & x > b^d \end{cases}$$

Expected Value: The expected value of a loguniform  $(c,d)$  random variable is

$$E(X) = \frac{b^d - b^c}{(d - c) \ln(b)}$$

Variance: The variance of a loguniform  $(c,d)$  random variable is

$$V(X) = \frac{b^{2d} - b^{2c}}{2(d - c) \ln(b)} - \left[ \frac{b^d - b^c}{(d - c) \ln(b)} \right]^2$$

Median: The median of a loguniform  $(c,d)$  random variable distribution is  $b^{(c+d)/2}$ .

#### 45.5.4 Algorithms for the Triangular Distribution

Generation Algorithm: The triangular distribution has probability density function:

$$f(x) = \begin{cases} 2(x - a)/[(b - a)(c - a)] & \text{for } a < x \leq b \\ 2(c - x)/[(c - b)(c - a)] & \text{for } b \leq x < c \end{cases}$$

and takes the value 0 elsewhere.

The following equation provides the generation algorithm for the triangular distribution:

$$F^{-1}(u) = \begin{cases} a + \sqrt{u(c - a)(b - a)} & \text{for } 0 \leq u \leq (b - a)/(c - a) \\ c - \sqrt{(1 - u)(c - a)(c - b)} & \text{for } (b - a)/(c - a) \leq u \leq 1 \end{cases}$$

**Cumulative Distribution Function:** The CDF algorithm for the triangular distribution takes the following form:

$$F(x) = \begin{cases} 0 & x \leq a \\ \frac{(x-a)^2}{(c-a)(b-a)} & a < x \leq b \\ 1 - \frac{(x-c)^2}{(c-a)(c-b)} & b < x \leq c \\ 1 & x \geq c \end{cases}$$

**Expected Value:** The expected value of a random variable with the triangular distribution is  $(a+b+c)/3$ .

**Variance:** The variance of a random variable with the triangular distribution is

$$V(X) = \left[ \frac{a^4c - a^4b + ab^4 - ac^4 - b^4c + bc^4}{6(b-a)(c-a)(c-b)} \right] - \left[ \frac{a+b+c}{3} \right]^2$$

**Median:** The median of a random variable with the triangular distribution depends on the location of the mode  $b$ . The median is:

$$M(X) = \begin{cases} a + \sqrt{(b-a)(c-a)/2} & \text{for } b \leq (a+c)/2 \\ c - \sqrt{(c-b)(c-a)/2} & \text{for } b > (a+c)/2 \end{cases}$$

#### 45.5.5 Algorithms for the Normal Distribution

**Generation Algorithm:** A normal  $(\mu, \sigma^2)$  random deviate,  $y$ , is obtained by generating a  $N(0,1)$  deviate,  $x$ , and then transforming that value using the equation  $y = \mu + \sigma x$ . The normally distributed random variable with mean  $\mu$  and variance  $\sigma^2$ , denoted as  $N(\mu, \sigma)$ , has the probability density function:

$$f(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

for  $-\infty < x < \infty$ ,  $-\infty < \mu < \infty$  and  $\sigma > 0$ .

The inverse cumulative distribution function for the  $N(0,1)$  random variable does not have a closed form expression. It is approximated by:

$$F^{-1}(p) = \begin{cases} q A(q^2)/B(q^2) & \text{for } |q| < 0.42 \\ \text{sgn}(q)C(r)/D(r) & \text{otherwise} \end{cases}$$

where  $q = p - 0.5$ , and  $r = \sqrt{\ln(0.5 - |q|)}$ . The quantity  $(0.5 - |q|)$  is formed as  $p$  or, to avoid cancellation if  $p$  is small, as  $(1-p)$ . The letters A, B, C, and D represent polynomials of order 3, 4, 3, and 2, respectively, whose coefficients are given in Beasley and Springer (1985), and  $\text{sgn}(q) = 1$  if  $q > 0$  and  $-1$  if  $q < 0$ .

**Cumulative Distribution Function:** There is no closed form analytic expression for the CDF of a random variable with the normal distribution. Algorithm AS 66 (Hill 1985) can be used to generate an accurate numerical approximation.

**Expected Value:** The expected value of a random variable with the  $N(\mu, \sigma^2)$  distribution is  $\mu$ .

**Variance:** The variance of a random variable with the  $N(\mu, \sigma^2)$  distribution is  $\sigma^2$ .

**Median:** The median of a random variable with the  $N(\mu, \sigma^2)$  distribution is  $\mu$ .

### 45.5.6 Algorithms for the Lognormal Distribution

**Generation Algorithm:** The logarithm of a random variable that is lognormally distributed is distributed as a normal  $N(\mu, \sigma^2)$  random variable (thus the name lognormal). The probability density function of the lognormal distribution is:

$$f(x) = \frac{A}{x\sigma\sqrt{2\pi}} e^{-[\log(x)-\mu]^2/2\sigma^2}$$

for  $x>0$  and  $\sigma>0$ . Because this distribution is available in both base 10 and natural logarithm base form, the constant A is  $1/\log_e 10$  for base 10 and 1 for the natural logarithm base. The logarithm  $\log(x)$  is also evaluated in terms of the chosen base.

A lognormal random variable,  $x$ , is generated using a two-step process. First, a value,  $y$ , is generated from the  $N(\mu, \sigma^2)$  distribution. This value is then used in the expression  $x=b^y$ , where the base  $b$  is either 10 or the natural constant  $e$ , as desired.

**Cumulative Distribution Function:** There is no closed form analytic expression for the CDF of a random variable with the lognormal distribution. However, a closed form expression can be accomplished by generating the CDF of the logarithm of the random variable. Algorithm AS 66 (Hill 1985) can be used to generate an accurate numerical approximation.

**Expected Value:** The expected value of a random variable with the  $\text{lognormal}(\mu, \sigma^2)$  distribution is

$$E(X) = e^{(\mu+0.5\sigma^2)}$$

**Variance:** The variance of a random variable with the  $\text{lognormal}(\mu, \sigma^2)$  distribution is

$$V(X) = e^{\sigma^2} \left( e^{\sigma^2} - 1 \right) e^{2\mu}$$

**Median:** The median of a random variable from the  $\text{lognormal}(\mu, \sigma^2)$  distribution is  $e^\mu$ .

### 45.5.7 Algorithms for the User-Defined Distribution

**Generation Algorithm:** In addition to selecting from parametric families of distributions, the user may implement any other distribution by supplying a table of data pairs corresponding to the pairs  $[F(x), x]$ . Thus, the user provides the SAC code with discrete evaluations of the cumulative distribution function.

The algorithm linearly interpolates between these points to solve for  $F^{-1}$  when generating a random deviate. The interpretation of the table of data is that the CDF takes on values on a continuum of values rather than a discrete set of values.

### 45.5.8 Algorithms for the Beta Distribution

Generation Algorithm: The beta random variable,  $x$ , is described by the probability density function:

$$f(x) = \frac{x^{p-1}(1-x)^{q-1}}{B(p,q)}$$

for  $p>0$ ,  $q>0$ , and  $0<x<1$ . This variable can be transformed to the interval  $(a,b)$ , and the resulting probability density function for the random variable,  $y$ , takes the form:

$$f(x) = \frac{(b-a)^{-(p+q+1)}(y-a)^{p-1}(b-y)^{q-1}}{B(p,q)}$$

for  $p>0$ ,  $q>0$ , and  $a<y<b$ . The second expression for the probability density function can be obtained from the first by the change of variable  $y=(b-a)x+a$ .

A closed form expression for the beta inverse cumulative distribution function does not exist. The algorithm implemented is provided in Algorithm AS 64/AS 109 (Griffiths and Hill 1985, p. 121). Algorithm AS 64/AS 109 requires the logarithm of  $B(p,q)$ . Using the relationship between the beta and gamma functions (Mood et al. 1974, p. 535),  $B(p,q)=\Gamma(p+q)/\{\Gamma(p)\Gamma(q)\}$ , where  $\Gamma(\cdot)$  denotes the gamma function, the logarithm of  $B(p,q)$  is computed using Algorithm ACM 291 (Pike and Hill 1966). Algorithm AS 64/AS 109 uses approximate starting values and a Newton-Rhapson iterative method to achieve a final solution.

Cumulative Distribution Function: There is no closed form analytic expression for the CDF of a random variable with the beta distribution. Therefore, the numerical approximation specified in Algorithm AS 63 (Mujamder and Bhattacharjee 1985) is used. This algorithm is based on the reduction method first published by Soper (1921).

Expected Value: The expected value of a random variable with the Beta distribution is  $a+(b-a)p/(p+q)$

$$E(X) = a + (b-a) \frac{p}{p+q}$$

Variance: The variance of a random variable with the Beta distribution is

$$V(X) = \frac{pq(b-a)^2}{(p+q+1)(p+q)^2}$$

Median: No analytically tractable expression is available for the median of a random variable with the Beta distribution.

### 45.5.9 Algorithms for the Log Ratio Distribution

**Generation Algorithm:** Let  $y$  denote a random variable from the log ratio distribution on the interval  $(a,b)$ . The probability density function for  $y$  is:

$$f(y) = \frac{(b-a)e^{-(\ln\{(y-a)/(b-y)\}-\mu)^2/(2\sigma^2)}}{\sigma(y-a)(b-y)\sqrt{2\pi}}$$

where  $a < b$  and  $a < y < b$ .

A random variable,  $y$ , from the log ratio distribution is generated using a two-step process. First, a value,  $x$ , is generated from the  $N(\mu, \sigma^2)$  distribution. This value is then used in the expression:

$$F^{-1}(x) = \frac{a + be^x}{1 + e^x}$$

**Cumulative Distribution Function:** There is no closed form analytical expression for the CDF of a random variable with the log ratio distribution.

**Expected Value:** There is no closed form analytical expression for the expected value of a random variable with the log ratio distribution.

**Variance:** There is no closed form analytical expression for the variance of a random variable with the log ratio distribution.

**Median:** There is no closed form analytical expression for the median of a random variable with the log ratio distribution.

### Algorithms for the Hyperbolic Arcsine Distribution

**Generation Algorithm:** Let the random variable  $x$  be a normally distributed random variable with mean  $\mu$  and variance  $\sigma^2$ . Then, let  $y$  be a random variable defined as  $y = \sinh^{-1}(x)$ . The probability density function for  $y$  is:

$$f(y) = \frac{0.5(e^y + e^{-y})e^{-(\sinh(y)-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

for  $-\infty < y < \infty$ ,  $-\infty < \mu < \infty$  and  $\sigma^2 > 0$ .

A random variable,  $y$ , from the hyperbolic arcsine distribution is generated using a two-step process. First, a value,  $x$ , is generated from the  $N(\mu, \sigma^2)$  distribution. This value is used in the expression:

$$F^{-1}(x) = \sinh(x)$$

Cumulative Distribution Function: There is no closed form analytical expression for the CDF of a random variable with the hyperbolic arcsine distribution.

Expected Value: There is no closed form analytical expression for the expected value of a random variable with the hyperbolic arcsine distribution.

Variance: There is no closed form analytical expression for the variance of a random variable with the hyperbolic arcsine distribution.

Median: There is no closed form analytical expression for the median of a random variable with the hyperbolic arcsine distribution.

## 46.0 References

- Beasley JD and SG Springer. 1985. "Algorithm AS 111, The Percentage Points of the Normal Distribution" in *Applied Statistics Algorithms*, P Griffiths and ID Hill (eds.). Ellis Horwood Limited, Chichester, England.
- DOE – U.S. Department of Energy. 1998. *Screening Assessment and Requirements for a Comprehensive Assessment, Columbia River Comprehensive Assessment*. DOE/RL-96-16, Rev 1, Final, U.S. Department of Energy, Richland, Washington. Accessed September 15, 2006 at <http://sesp.pnl.gov/Reports/CRCIA/reports.htm>.
- DOE – U.S. Department of Energy. 1999a. *Radioactive Waste Management Manual*. DOE M 435.1-1, U.S. Department of Energy, Washington, D.C. Accessed September 15, 2006 at <http://www.directives.doe.gov/pdfs/doe/doetext/neword/435/435.htm>.
- DOE – U.S. Department of Energy. 1999b. *Implementation Guide for Use with DOE M 435.1-1*. DOE G 435.1-1, U.S. Department of Energy, Washington, D.C. Accessed September 15, 2006 at <http://www.directives.doe.gov/pdfs/doe/doetext/neword/435/435.htm>.
- DOE Order 435.1. 2001. *Radioactive Waste Management*. U.S. Department of Energy, Washington, D.C. Accessed September 15, 2006 at <http://www.directives.doe.gov/pdfs/doe/doetext/neword/435/o4351c1.html>.
- EPA – U.S. Environmental Protection Agency. 1998. *Guidelines for Ecological Risk Assessment*. EPA/630/R-95/002F, Risk Assessment Forum, U.S. Environmental Protection Agency, Washington, D.C. Also published in the Federal Register 63(93):26846-26924.
- Eslinger, PW, DW Engel, LH Gerhardstein, CA Lopresti, TB Miley, WE Nichols, DL Streng, and SK Wurster. 2006a. Updated User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes. Volume 1: Inventory, Release, and Transport Modules. PNNL-16115, Volume 1, Pacific Northwest National Laboratory, Richland, Washington.
- Eslinger PW, TB Miley, and C Arimescu. 2006b. *Updated User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes. Volume 2: Impact Modules*. PNNL-16115, Volume 2, Pacific Northwest National Laboratory, Richland, Washington.
- Firestone M, P Fenner-Crisp, T Barry, D Bennet, S Chang, M Callahan, A Burke, J Michaud, M Olsen, P Cirone, D Barnes, WP Wood, and SM Knott. 1997. *Guiding Principles for Monte Carlo Analysis*. EPA/630/R-97/001, Risk Assessment Forum, U.S. Environmental Protection Agency, Washington, D.C.
- Fishman GS and LR Moore. 1986. "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus  $2^{31}-1$ " in *Siam Journal of Scientific and Statistical Computing* 7(1):24-44.
- Freedman, VL, Y Chen, A Gilca, CR Cole, and SK Gupta. 2006. *CFEST Coupled Flow, Energy & Solute Transport Version CFFEST005 User's Guide*. PNNL-15915 Rev. 1, Pacific Northwest National Laboratory, Richland, Washington.

- Griffiths P and ID Hill (eds.). 1985. *Applied Statistics Algorithms*. Ellis Horwood Limited, Chichester, England .
- Hill ID. 1985. "Algorithm AS 66, The Normal Integral." In *Applied Statistics Algorithms*, eds P Griffiths and ID Hill, Ellis Horwood Limited. Chichester, England.
- Hoare CAR. 1961. "Partition: Algorithm 63," "Quicksort: Algorithm 64," and "Find: Algorithm 65." In *Comm. ACM* 4:321-322.
- Iman RL and WJ Conover. 1982. "A Distribution-free Approach to Inducing Rank Correlations Among Input Variables." In *Communications in Statistics* B11(3):311-334.
- Lewis PA, AS Goodman, and JM Miller. 1969. "A Pseudo-random Number Generator for the System/360." In *IBM Systems Journal* 8(2):136-145.
- Mood AM, FA Graybill, and DC Boes. 1974. *Introduction to the Theory of Statistics*, Third Edition. McGraw-Hill Book Co., New York.
- Mualem Y. 1976. "A New Model For Predicting The Hydraulic Conductivity Of Unsaturated Porous Media." *Water Resources Research* 12(3):513-522.
- Mujamder, KL and GP Bhattacharjee. 1985. "Algorithm AS 64, Inverse of the Incomplete Beta Function Ratio." In *Applied Statistics Algorithms*, eds P Griffiths and ID Hill, Ellis Horwood Limited, Chichester, England.
- Park SK and KW Miller. 1988. "Random Number Generators: Good Ones are Hard to Find" in *Comm. of the ACM* 31(10):1192-1201.
- Perkins WA and MC Richmond. 2004a. *MASS2, Modular Aquatic Simulation System in Two Dimensions, Theory and Numerical Methods*. PNNL-14820-1. Pacific Northwest National Laboratory, Richland, Washington.
- Perkins WA and MC Richmond. 2004b. *MASS2, Modular Aquatic Simulation System in Two Dimensions, User Guide and Reference*. PNNL-14820-2. Pacific Northwest National Laboratory, Richland, Washington.
- Pike MC and ID Hill. 1966. "ACM Algorithm 291: Logarithm of Gamma Function." In *Communications of the ACM* 9(9):684.
- Ramsdell Jr., JV and JP Rishel. 2006. *Regional Atmospheric Transport Code for Hanford Emission Tracking, Version 2 (RATCHET2): Modification and Implementation of RATCHET for Use in SAC*. PNNL-16071. Pacific Northwest National Laboratory, Richland, WA.
- Simpson BC, RA Corbin, and SF Agnew. 2001. *Hanford Soil Inventory Model*. BHI-01496 Rev 0., Bechtel Hanford, Inc., Richland, Washington.
- Soper, H.E., 1921. "The numerical evaluation of the incomplete beta-function." In *Tract for Computers* No. 7, Cambridge: Cambridge University Press.



van Genuchten MT. 1980. "A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils." *Soil Sci. Soc. Am. J.* 44:892-898.

White MD and M Oostrom. 2000. *STOMP Subsurface Transport Over Multiple Phases User's Guide, Version 2.0*. PNNL-12034, Pacific Northwest National Laboratory, Richland, Washington.

# **Distribution**

## **U.S. Department of Energy, RL**

B. L. Charboneau (CD)	A6-33
R. D. Hildebrand (CD)	A5-13
J. G. Morse (CD)	A6-38
DOE Public Reading Room (2P)	H2-53

## **U.S. Department of Energy, ORP**

R. W. Lober (CD)	H6-60
------------------	-------

## **CH2M HILL**

F. J. Anderson (CD)	E6-35
F. Mann (CD)	E6-35

## **Fluor Federal Services**

R. Puigh (CD)	E6-17
---------------	-------

## **Fluor Hanford, Inc.**

T. W. Fogwell (CD)	E6-35
--------------------	-------

## **Pacific Northwest National Laboratory**

C. Arimescu (1P)	K6-52
R. L. Aaberg (CD)	K3-54
M. P. Bergeron (CD)	K9-36
R. W. Bryce (CD/P)	E6-35
D. W. Engel (CD/P)	K6-08
P. W. Eslinger (5 CD/5P)	K6-52
V.L. Freedman (CD)	K9-36
B. A. Kanyid (CD)	K7-22
C. T. Kincaid (CD/P)	K9-33
C. A. Lopresti (CD)	K6-08
T. B. Miley (3CD/3P)	K6-52
B. A. Napier (CD)	K3-54
W. E. Nichols (CD/P)	K9-33
M. C. Richmond (CD)	K9-33
R. G. Riley (P)	K2-21
D. L. Strenge (CD)	K3-54
S. K. Wurstner (CD/P)	K9-36
Hanford Technical Library (2P)	P8-55